ABORDAGENS DE TÉCNICAS DE LSH APLICADAS AO PROBLEMA DE SIMILARIDADE DE DOCUMENTOS

Danielle Caled Vieira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Geraldo Bonorino Xexéo

Rio de Janeiro
Fevereiro de 2016

ABORDAGENS DE TÉCNICAS DE LSH APLICADAS AO PROBLEMA DE SIMILARIDADE DE DOCUMENTOS

Danielle Caled Vieira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof. Geraldo Zimbrao da Silva, D.Sc.

Prof. Marco Antonio Casanova, Ph.D

RIO DE JANEIRO, RJ – BRASIL
FEVEREIRO DE 2016

Vieira, Danielle Caled

Abordagens de técnicas de LSH aplicadas ao problema de similaridade de documentos/Danielle Caled Vieira. – Rio de Janeiro: UFRJ/COPPE, 2016.

XI, 85 p.: il.; 29, 7cm.

Orientador: Geraldo Bonorino Xexéo

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2016.

Referências Bibliográficas: p. 57 – 62.

1. Recuperação de informação. 2. Locality-Sensitive Hashing. 3. Minwise Hashing. I. Xexéo, Geraldo Bonorino. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha avó Yvonne.

Agradecimentos

Gostaria de agradecer aos meus pais, Jair e Lúcia, por toda educação que me ofereceram e por todo amor que me dedicaram. Obrigada por serem meus amigos e companheiros, por se prontificaram a me ajudar de todas as formas, não medido esforços para isso. Agradeço também por terem me ensinado os valores que me transformaram na pessoa que hoje sou.

Agradeço à minha avó Yvonne, por seu carinho e dedicação, estando sempre ao meu lado, me apoiando e incentivando não só na minha trajetória acadêmica, como em toda a minha vida.

À minha irmã Flavia, cuja presença sempre me proporcionou momentos agradáveis, através de diversas conversas construtivas e, sem dúvidas, fundamentais ao meu desenvolvimento pessoal.

Ao meu amigo Miguel Wolf que me incentivou e ajudou a concluir este projeto, e sem os qual não teria alcançado sucesso. Obrigada pela paciência, por me manter motivada e pela ajuda com a revisão.

A todos os amigos que conheci durante este período na COPPE e COPPETEC, em especial ao Pedro Beyssac, pelos diversos trabalhos que realizamos juntos ao longo do mestrado e ao Fellipe Duarte, que sempre se dispôs a me ajudar, me auxiliando desde a concepção até a conclusão dessa dissertação.

Agradeço ao meu orientador, Prof. Geraldo Xexéo, e aos demais professores do PESC, por me proporcionarem as bases e os conhecimentos necessários à realização desse trabalho.

E, finalmente, aos meus amigos, que ainda que não tenham participado diretamente da minha vida acadêmica, sempre estiveram ao meu lado e souberam entender as minhas ausências em tantas reuniões desmarcadas ou adiadas.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ABORDAGENS DE TÉCNICAS DE LSH APLICADAS AO PROBLEMA DE SIMILARIDADE DE DOCUMENTOS

Danielle Caled Vieira

Fevereiro/2016

Orientador: Geraldo Bonorino Xexéo

Programa: Engenharia de Sistemas e Computação

Neste trabalho, são apresentadas quatro novas abordagens para o cálculo da similaridade entre conjuntos de alta dimensionalidade através da estimativa do tamanho relativo da sua interseção. Esses novos métodos foram desenvolvidos a partir da abordagem *Minwise Hashing*, uma instância da família de funções *Locality-Sensitive Hashing*.

O foco do nosso estudo é explorar diferentes formas de representar documentos em grandes *corpora*. Cada uma das abordagens propostas examina uma ou mais características (*operadores*) dos documentos analisados. E, a partir delas, é possível estimar a similaridade entre pares de documentos, extraindo informações úteis para diferentes cenários.

Além das abordagens propostas, também apresentamos os experimentos realizados em uma aplicação real de reúso textual, o *corpus* METER, constituído de publicações jornalísticas da imprensa britânica. Por fim, comparamos os resultados dos experimentos aos produzidos pelo método *Minwise Hashing*, e constatamos que a abordagem proposta *MinMaxwise Hashing* apresenta resultados superiores aos obtidos com os outros métodos estudados no presente trabalho.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

LSH APPROACHES APPLIED TO DOCUMENT SIMILARITY PROBLEM

Danielle Caled Vieira

February/2016

Advisor: Geraldo Bonorino Xexéo

Department: Systems Engineering and Computer Science

In this work, we present four new approaches for calculating the similarity between high dimensional sets by estimating the relative size of their intersection. These new methods have been developed from the *Minwise Hashing* approach, an instance of the family of functions *Locality-Sensitive Hashing*.

The focus of our study is to explore different ways of representing documents in large corpora. Each of the proposed approaches examines one or more characteristics (*operators*) of the analyzed documents. And from them, it is possible to estimate the similarity between pairs of documents, extracting useful information for different scenarios.

In addition to the proposed approaches, we also present the experiments made in a real application of textual reuse, the METER corpus, consisting of journalistic publications of the British press. Finally, we compare the experimental results to those produced by the method *Minwise Hashing*, and we verify that the proposed approach *MinMaxwise Hashing* provides results superior to those obtained with the other methods studied in this work.

Sumário

Lista de Figuras **x**

Lista de Tabelas **xi**

1 Introdução **1**

1.1 Motivação e Problema 1

1.2 Objetivo 2

1.3 Estrutura da Dissertação 3

2 Relacionamentos Textuais e Reúso de Texto **4**

2.1 Co-derivação 8

2.2 Reúso Jornalístico 9

3 Heurísticas de Similaridade no Reúso de Texto **13**

3.1 Introdução aos modelos adotados em tarefas de identificação de reúso 13

3.2 Algoritmos Aleatórios 16

3.3 *Fingerprinting* 17

3.3.1 *Fingerprinting* Completo 19

3.3.2 *Fingerprinting* Seletivo 20

3.4 *Locality-Sensitive Hashing* 20

3.5 *Minwise Hashing* 23

3.5.1 *Shingling* 24

3.5.2 Definição do Problema da Interseção de Conjuntos e Estimativa da Similaridade 25

3.5.3 Representação Matricial 26

3.5.4 Implementação do Algoritmo *min-Hash* 27

4 Métodos Propostos **30**

4.1 Semi-Reticulados e Reticulados 30

4.2 Operadores de Agregação 32

4.3 Variantes do método *Minwise Hashing* 33

4.3.1 *Maxwise Hashing* 34

4.3.2 *MinMinwise Hashing e MaxMaxwise Hashing* 36
4.3.3 *MinMaxwise Hashing* 37
4.4 Formalização dos métodos propostos 39
4.4.1 *Framework* de aplicação de famílias de técnicas LSH 39
4.4.2 Análise de complexidade dos métodos propostos 44

5 Resultados e Discussões **46**
5.1 Metodologia para avaliação dos métodos propostos 46
5.2 Métricas de Avaliação 47
5.3 Precisão Interpolada 47
5.4 Experimentos 49

6 Conclusões e Trabalhos Futuros **55**

Referências Bibliográficas **57**

A Abordagens baseadas em Recuperação da Informação para detecção de reúso de texto **63**

Lista de Figuras

2.1 Possíveis casos de co-derivação de documentos. 8

3.1 LSH: Objetos próximos são mapeados para o mesmo *slot*. 21

4.1 Fluxograma proposto 39
4.2 Representação gráfica das permutações π_1 e π_2 aplicadas aos conjuntos C_1 e C_2. 41
4.3 Assinaturas dos conjuntos C_1 e C_2 de acordo com (a) a função de permutação π_1 e (b) a função de permutação π_2. 42

5.1 Precisão em 11 níveis de revocação. 48
5.2 Precisão média em 11 níveis de revocação. 49
5.3 Área sob a curva de precisão média em 11 níveis de revocação ao utilizar o índice de Jaccard. 51
5.4 Área sob a curva de precisão média em 11 níveis de revocação ao utilizar a similaridade do cosseno. 51
5.5 Área sob a curva de precisão média em 11 níveis de revocação ao utilizar o coeficiente de Dice 52
5.6 Precisão média em 11 níveis de revocação para 1024 dimensões. Experimentos usando as métricas: Similaridade do Cosseno, Índice de Jaccard e Coeficiente de Dice. 53

Lista de Tabelas

2.1 Exemplo de reúso de texto jornalístico. Ambas as matérias foram publicadas no dia 2 de outubro de 2014. 6

3.1 Exemplo de construção da representação matricial dos conjuntos C_1, C_2 e C_3. 27
3.2 Permutação das linhas da matriz característica M dos conjuntos C_1, C_2 e C_3. 27
3.3 Matriz assinatura dos conjuntos C_1, C_2 e C_3 dadas as permutações produzidas pelas funções *hash* h_1, h_2, h_3. 29

4.1 Exemplos de $O_k \subset O$. 34
4.2 Ordens induzidas pelas permutações π_1 e π_2. 41
4.3 Construção da matriz de assinaturas S para N permutações de I conjuntos através de um conjunto de *operadores* O_k, com P *operadores*. 44

5.1 Precisão interpolada das consultas c_1 e c_2. 48

Capítulo 1

Introdução

1.1 Motivação e Problema

Problemas na área de Recuperação da Informação, como a identificação do reúso de texto, são um tema de considerável interesse teórico e prático que vem sendo bastante estudado e discutido (CLOUGH *et al.*, 2002a). Por vezes, ao realizar uma tarefa de Recuperação da Informação em um grande conjunto de textos, é necessário procurar por algum elemento que possa indicar um caso suspeito (BARRÓN-CEDEÑO, 2010). Uma das principais formas de resolver este problema é através da medição da similaridade entre documentos (BARRÓN-CEDEÑO, 2010). Diferentes abordagens são empregadas para encontrar trechos de textos parecidos ou mesmo iguais em grandes volumes de dados, sendo uma delas o uso de *fingerprints* (HEINTZE *et al.*, 1996). A técnica *Fingerprinting* (MANBER *et al.*, 1994) realiza o mapeamento de uma *string* para um número inteiro aleatório (HEINTZE *et al.*, 1996), facilitando a representação e manipulação do conteúdo de documentos.

Além da abordagem *Fingerprinting*, também se pode utilizar o conjunto de técnicas *Locality-Sensitive Hashing* (LSH) para a redução da dimensionalidade de documentos em bases muito extensas (BUHLER, 2001). O uso dessa família de técnicas é vantajoso para a diminuição do tempo computacional em problemas que lidam com representações esparsas de documentos.

Uma das possíveis abordagens baseadas em LSH é o algoritmo *Minwise Hashing*, desenvolvido por BRODER (1997, 2000), BRODER *et al.* (1998). Este algoritmo foi criado com a finalidade de tratar problemas de similaridade de documentos através de um mecanismo de busca, produzido e utilizado pela AltaVista, para detectar páginas *web* duplicadas.

Ao trabalhar com grandes coleções de páginas *web*, BRODER (2000) se preocupa com a alta dimensionalidade dos documentos e com a eficiência da sua indexação. Assim, ele deduz que uma amostragem dos itens da coleção serviria aos propósitos

do seu mecanismo de busca. Em contrapartida, no entanto, os valores calculados para a similaridade entre dois documentos corresponderiam a um valor aproximado.

BRODER (2000) baseou-se em dois aspectos para o cálculo da similaridade entre documentos: "primeiro, a similaridade é vista como um problema de interseção de conjuntos (de *strings*), e, segundo, o tamanho relativo das interseções é avaliado por um processo de amostragem aleatória que pode ser feito de forma independente para cada documento". PAGH *et al.* (2014) aprofundam o estudo de Broder, demonstrando ser possível utilizar os k menores valores gerados por uma única permutação para sumarizar um conjunto, reduzindo, assim, o número de *bits* necessários a esta representação.

Neste trabalho, serão apresentadas quatro novas famílias de técnicas de *Locality-Sensitive Hashing*, inspiradas no algoritmo *Minwise Hashing*. Pretendemos, com isso, estudar o problema do cálculo da similaridade entre conjuntos de alta dimensionalidade através da estimativa do tamanho relativo da sua interseção. Portanto, tendo por base o trabalho de BRODER (2000) e PAGH *et al.* (2014), adotaremos como metodologia para a implementação das abordagens propostas a exploração de diferentes características para representar a interseção. Estudaremos os algoritmos aplicados a uma conhecida base de reúso de texto jornalístico, o *corpus* METER (CLOUGH *et al.*, 2002b, GAIZAUSKAS *et al.*, 2001). De posse dos resultados, poderemos, também, avaliar os métodos estudados comparando-os ao *Minwise Hashing*.

1.2 Objetivo

Esta dissertação apresenta quatro novas abordagens, o *Maxwise Hashing*, o *MinMaxwise Hashing*, o *MinMinwise Hashing* e o *MaxMaxwise Hashing*. Tais métodos, assim como o *Minwise Hashing*, são instâncias da família de funções *Locality-Sensitive Hashing* (CHARIKAR, 2002), cujo propósito é a redução da dimensionalidade em problemas com grandes volumes de dados.

Tal como BRODER (1997), pretendemos tratar a similaridade de dois conjuntos através do problema da interseção. Porém, nosso objetivo consiste em estudar diferentes *operadores* que possam representá-la. Assim, enquanto o algoritmo *Minwise Hashing* utiliza uma série de transformações que lhe permitem analisar o menor elemento que representa a interseção de dois conjuntos, o *Maxwise Hashing* tem por objetivo usar as mesmas transformações aplicadas aos conjuntos para extrair e analisar o seu máximo. Espera-se, portanto, que ambos os métodos apresentem resultados parecidos, pois utilizam a mesma quantidade de informação e exploram os limites que definem a interseção de dois conjuntos e, logo, a sua similaridade.

Os métodos *MinMinwise Hashing* e *MaxMaxwise Hashing*, inspirados na ideia de PAGH *et al.* (2014), buscam examinar a similaridade entre dois conjuntos por

meio de dois *operadores* da sua interseção, sendo eles seus dois menores elementos ou seus dois maiores elementos, respectivamente. A abordagem *MinMaxwise Hashing*, assim como os métodos anteriores, tem por objetivo utilizar transformações aplicadas aos conjuntos para, então, analisar a sua similaridade. Novamente estudaremos a interseção de dois conjuntos, entretanto, nesse método, buscaremos compreendê-la através de dois *operadores* (o mínimo e o máximo) provenientes das transformações realizadas. Com esses métodos foi possível obter resultados similares (no caso do *MinMinwise Hashing* e do *MaxMaxwise Hashing*) ou melhores (caso do *MinMaxwise Hashing*) em relação às abordagens *Minwise Hashing* e *Maxwise Hashing*, visto que a mesma quantidade de dados proveniente das transformações nos permite trabalhar com o dobro da informação das técnicas, levando, assim, a um ganho linear.

Ao longo do presente trabalho, buscaremos apresentar as abordagens propostas, com sua fundamentação teórica e seus algoritmos, além de aplicar as técnicas em um caso real de reúso de texto jornalístico. Avaliaremos as novas abordagens com base no seu resultado ao identificar quais documentos são relevantes para a consulta executada. Nossas consultas pretendem testar a capacidade dos métodos comparados em identificar quais documentos no *corpus* METER são parcialmente ou totalmente derivados de notícias publicadas pela agência de notícias britânica Press Association.

1.3 Estrutura da Dissertação

Essa dissertação está organizada da seguinte forma: o Capítulo 1) descreve a motivação, o problema que pretendemos tratar e os objetivos da dissertação. O Capítulo 2 apresenta uma revisão bibliográfica acerca de relacionamentos textuais e reúso de texto. O Capítulo 3 aborda as heurísticas de similaridade no reúso de texto, incluindo uma explicação mais aprofundada do algoritmo proposto por BRODER (1997), o *Minwise Hashing*. As novas abordagens estudadas nesse trabalho (*Maxwise Hashing*, *MinMaxwise Hashing*, *MinMinwise Hashing* e *MaxMaxwise Hashing*) são explicadas no Capítulo 4. A apresentação dos resultados e sua discussão são realizadas no Capítulo 5. Concluímos o presente trabalho no Capítulo 6.

Por fim, apresentamos no Apêndice A o relatório desenvolvido durante o estágio realizado no Institut de Recherche en Informatique de Toulouse ao longo do curso de mestrado. Este relatório se refere a um estudo de abordagens clássicas baseadas em Recuperação da Informação para a detecção de reutilização de texto, com experimentos também desenvolvidos sobre o *corpus* METER.

Capítulo 2

Relacionamentos Textuais e Reúso de Texto

O reúso de textos, conceitos e conteúdos é um método de produção muito explorado e frequentemente estudado, como em (BARRÓN-CEDEÑO, 2010, BENDERSKY e CROFT, 2009, CLOUGH, 2010, CLOUGH *et al.*, 2002a,b, LEVY, 1993, WILKS, 2004). Presente tanto em antigas narrativas de histórias quanto em documentos escritos em papel, a atividade de reúso aumentou consideravelmente com o uso da tecnologia digital como ferramenta de escrita (CLOUGH, 2010, LEVY, 1993, WILKS, 2004). Outros avanços na tecnologia da informação, como o acesso à Internet e o uso de ferramentas de busca, também contribuem para a reutilização de texto em diversos níveis de modificação (CLOUGH, 2010). Alguns exemplos de reúso incluem a criação de textos literários e históricos, tradução, revisão e resumo de textos existentes (CLOUGH *et al.*, 2003).

CLOUGH *et al.* (2003) definem reúso de texto como uma "atividade em que um material escrito pré-existente é reutilizado durante a criação de um novo texto, intencionalmente ou não-intencionalmente". A reutilização de texto envolve um processo em que um autor reescreve ou edita, com ou sem permissão do dono, um dado documento (CLOUGH, 2010). Motivado pela dificuldade encontrada no desenvolvimento de uma nova ideia, é muito provável que esse autor se baseie em obras anteriores relacionadas ao seu tema de estudo para redigir seu trabalho (BARRÓN-CEDEÑO, 2010).

De acordo com CLOUGH *et al.* (2002b), é importante perceber que o reúso de texto se trata de um fenômeno contínuo que se estende da reutilização literal, palavra por palavra (*verbatim*), passando por "diferentes graus de transformação que envolvem substituições, inserções, exclusões e reordenações, até uma situação em que o texto tenha sido gerado de forma totalmente independente, mas onde os mesmos eventos sejam descritos por um outro membro da mesma comunidade linguística e cultural (e, portanto, onde se pode antecipar sobreposições de vários

tipos)".

A expressão "reúso de texto" abrange uma série de transformações textuais, como sínteses, reproduções exatas, reformulações de informações provenientes de outras fontes e relatórios (que têm pouco em comum com o documento original, exceto o assunto) (BENDERSKY e CROFT, 2009). As transformações variam de adições de novos fatos ou opiniões ao texto da fonte, exclusão de trechos originais, alterações sutis na escrita, completa reformulação do texto, mudanças no estilo de escrita para atender a uma demanda diferenciada, simplificação, tradução de um texto original para um outro idioma, entre outros processos de manipulação de texto (BARRÓN-CEDEÑO, 2010, BENDERSKY e CROFT, 2009, CLOUGH *et al.*, 2003).

CLOUGH (2010) trata o problema do reúso de texto sob duas perspectivas, a do autor e a do leitor. A primeira, a perspectiva do autor, consiste na procura e edição do material do seu interesse (CLOUGH, 2010, LEVY, 1993). A segunda, a do leitor, é vista como um problema de análise ou atribuição de autoria em que, considerando-se dois textos, busca-se descobrir se um deles é derivado do outro (CLOUGH, 2010, WILKS, 2004).

Segundo WILKS (2004), o reúso é uma forma independente de atividade linguística em que métodos computacionais empregados para detectá-lo diferem sutilmente daqueles usados em problemas de plágio [1]. A identificação de reúso de texto pode se tornar uma tarefa complicada devido às diversas transformações a que um texto está sujeito, tais como a reutilização literal, ou casos mais complexos, de paráfrase e sumarização, que tornam a nova versão bem diferente da original (CLOUGH, 2010). Por esse motivo, torna-se evidente a subjetividade da avaliação em problemas de detecção do reúso (CLOUGH *et al.*, 2002b).

A prática do reúso aparece frequentemente no meio acadêmico. Estudantes são avaliados de acordo com a sua produção (publicações de artigos, teses, etc.), além de a atividade de pesquisa demandar esforço e tempo. A escassez destes recursos, aliada à incapacidade de interpretação de texto, falta de habilidade em desenvolver pesquisa e má compreensão do conceito de plágio, muitas vezes leva ao reúso de trechos de trabalhos anteriores inteiramente copiados. Ademais, outra atividade usualmente adotada por acadêmicos é o reúso de textos escritos por ele mesmo, mas para outros trabalhos.

Existem situações em que casos de reutilização são bem aceitos e considerados apropriados. Eles são vistos como casos benignos de adaptação, reescrita ou plágio, feitos em situações em que o autor do documento original não seja prejudicado (WILKS, 2004). Um exemplo no meio jornalístico consiste nas informações geradas

[1] O plágio se caracteriza quando da utilização de "ideias, conceitos ou frases de outro autor (que as formulou e as publicou), sem lhe dar o devido crédito, sem citá-lo como fonte de pesquisa." (NERY *et al.*, 2010)

por agências de notícias [2] que são vendidas a veículos de comunicação, como jornais, revistas, rádios e outras mídias para o seu reúso (CLOUGH *et al.*, 2002a, WILKS, 2004). Em troca de uma taxa paga às agências de notícias, a mídia interessada tem acesso ao conteúdo e pode utilizá-lo da forma que lhe for mais conveniente (seja, reproduzindo literalmente ou reescrevendo o texto de modo que ele seja compatível com o seu estilo e/ou interesse) (BARRÓN-CEDEÑO, 2010).

A Tabela 2.1 apresenta um exemplo de reúso de texto jornalístico. O texto original é proveniente de uma agência de notícias. Na segunda matéria, entretanto, o texto original foi adaptado e reformulado de acordo com as características de estilo e forma próprias à mídia que o publicou, ainda que conserve o mesmo conteúdo da fonte.

Fonte	**Trecho da Matéria**
Reuters Brasil* (Texto original)	**Quebra de protocolo de segurança pode ter levado à nova infecção por Ebola no Texas:** Uma quebra nos protocolos de segurança, possivelmente durante a remoção de equipamentos de proteção após o tratamento de um paciente com Ebola, pode ter causado a contração do vírus mortal por um profissional de saúde no Texas, disse um alto funcionário do setor de saúde dos Estados Unidos no domingo.
G1**	**Obama pede medidas para sistema de saúde se preparar para ebola:** Segundo informações divulgadas neste domingo, uma quebra nos protocolos de segurança, possivelmente durante a remoção de equipamentos de proteção após o tratamento do paciente com ebola, pode ter causado a contração do vírus mortal pela profissional de saúde.

*Trecho extraído de http://br.reuters.com/article/topNews/idBRKCN0I10NP20141012

**Trecho extraído de http://glo.bo/1qgi6sa

Tabela 2.1: Exemplo de reúso de texto jornalístico. Ambas as matérias foram publicadas no dia 2 de outubro de 2014.

Um outro meio no qual reúso de texto é bem aceito é a criação colaborativa de documentos. Além de projetos que costumam promover o desenvolvimento de manuais de *software*, existe ainda a escrita colaborativa de artigos, como no caso da Wikipédia [3] (BARRÓN-CEDEÑO, 2010). A Wikipédia consiste em um *framework* colaborativo de enciclopédia multilíngue que permite o reúso de conteúdo em artigos relacionados ou traduzidos (BARRÓN-CEDEÑO, 2010, WIKIPÉDIA, 2013). Uma

[2] As agências de notícias mais conhecidas são a Press Association e Reuters (ambas no Reino Unido) e a Associated Press e UPI (nos Estados Unidos) (WILKS, 2004). No Brasil, as principais agências de notícias são Agência Brasileira de Notícias (ABN News, uma das primeiras agências de notícias brasileiras), a Agência O Globo (criada para distribuir informações para os jornais O Globo, Extra e seus respectivos sites), Agência Estado (criada para dar suporte às mídias do grupo Estado) e a Agência Brasil (agência de notícias pública, criada para fortalecer o sistema público de comunicação).

[3] http://www.wikipedia.org

das grandes vantagens da Wikipédia é o estímulo ao referenciamento, especialmente no caso de reutilização de trechos de artigos (BARRÓN-CEDEÑO, 2010).

No contexto de reutilização textual, é importante definir os relacionamentos entre documentos. De acordo com cada tipo de relacionamento, é possível optar por uma abordagem mais apropriada que possa indicar o reúso. Não obstante, a busca exata por trechos de documentos, ainda que seja uma abordagem fácil de ser empregada, geralmente não é satisfatória (HEINTZE *et al.*, 1996). Como existem tipos mais elaborados de reúso, essa abordagem falha até mesmo em capturar trechos que tenham sido levemente modificados, ignorando relacionamentos textuais mais interessantes (HEINTZE *et al.*, 1996).

HEINTZE *et al.* (1996) sumarizam os tipos de relacionamentos entre documentos que podem ser considerados relevantes, a saber:

(a) Documentos idênticos.

(b) Documentos que são resultados de pequenas edições e/ou correções de outros documentos.

(c) Documentos que são reorganizações do conteúdo de outros documentos.

(d) Documentos que são revisões de outros documentos.

(e) Documentos que são versões condensadas ou estendidas de outros documentos (por exemplo, versões do mesmo artigo para periódicos e conferências).

(f) Documentos que incluem grandes porções de texto provenientes de outros documentos.

HEINTZE *et al.* (1996) observam ainda que, enquanto as cinco primeiras classes de relacionamentos podem ser identificadas com probabilidade bastante alta, para a classe restante, tolera-se um pequeno número de falsos-negativos e falsos-positivos.

BARRÓN-CEDEÑO (2010), por outro lado, organizou um conjunto de situações em que ocorre reutilização de texto. Ele tomou como base o trabalho de MARTIN (1994) e de MAURER *et al.* (2006). Ambos os autores identificam diversos métodos usados em plágio, mas BARRÓN-CEDEÑO (2010) selecionou alguns que reconhece, na verdade, como métodos de reúso de texto. A seguir, explica-se as situações nas quais se dá com maior frequência a reutilização:

(a) Reúso palavra por palavra (*verbatim*): é a forma de reúso mais óbvia e provável (MARTIN, 1994). Ocorre quando alguém copia fragmentos inalterados de um trabalho publicado sem que haja uso de aspas e/ou referência à fonte (BARRÓN-CEDEÑO, 2010, MARTIN, 1994). Esse tipo de reúso é o clássico "Copiar e Colar"(MAURER *et al.*, 2006).

(b) Paráfrase: Ocorre quando um conteúdo textual é copiado, ainda que se faça pequenas modificações. Algumas palavras do texto original podem ser mudadas ou o texto pode ser alterado ou parafraseado (BARRÓN-CEDEÑO, 2010, MARTIN, 1994).

(c) Reúso de ideias: Ocorre quando, embora não haja qualquer dependência em termos de palavras ou forma em relação à fonte, uma ideia original é reutilizada (MARTIN, 1994).

(d) Reúso de código fonte: Se dá com o uso de código de programação, algoritmos, classes, ou funções sem a devida citação do autor ou a sua permissão (BARRÓN-CEDEÑO, 2010, MAURER *et al.*, 2006).

(e) Reúso traduzido: Acontece quando há a tradução de algum conteúdo, ainda que com algumas modificações, sem que seja feita a referência ao texto original (BARRÓN-CEDEÑO, 2010, MAURER *et al.*, 2006).

2.1 Co-derivação

De acordo com BERNSTEIN e ZOBEL (2004), dois documentos são co-derivados se eles compartilham o mesmo conteúdo. Isto é, para que seja identificada uma co-derivação entre dois documentos, é necessário que algum trecho de um deles tenha sido derivado do outro, ou que um trecho de ambos tenha sido derivado de um terceiro documento (Figura 2.1) (BERNSTEIN e ZOBEL, 2004). A co-derivação é considerada uma forma de reúso, na qual também são incluídas revisões de documentos, resenhas e resumos (BARRÓN-CEDEÑO, 2010).

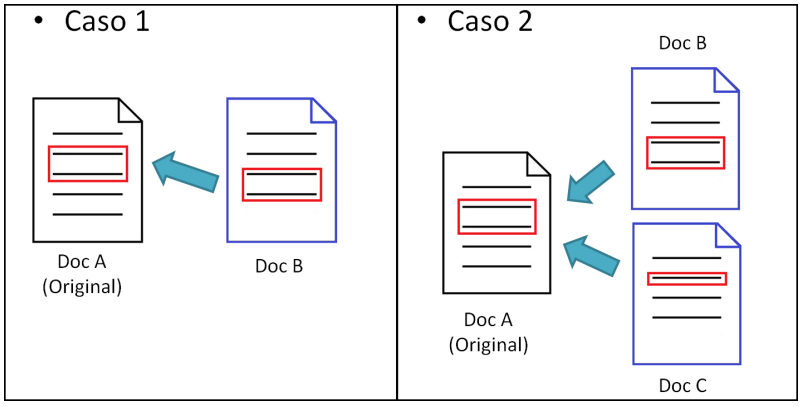

Figura 2.1: Possíveis casos de co-derivação de documentos.
Caso 1: Trecho de Doc B derivado de Doc A. Caso 2: Trechos de Doc B e Doc C derivados do documento original Doc A.

O conhecimento dos relacionamentos co-derivados entre documentos em uma coleção pode ser usado para produzir resultados mais informativos a partir de meca-

nismos de buscas, detecção de plágio e gerenciamento de versionamento de documentos em uma empresa (BERNSTEIN e ZOBEL, 2004). Ao se realizar análise de casos de co-derivação, assim como em casos de reutilização de texto e detecção de plágio, normalmente calcula-se a similaridade entre os documentos (BARRÓN-CEDEÑO, 2010, MAURER *et al.*, 2006).

Ainda que seja comum em casos de co-derivação a reprodução de longos blocos de texto com possíveis modificações intermitentes, não é necessário que, para serem considerados co-derivados, dois documentos utilizem as mesmas palavras, mesma estrutura ou o mesmo formato (HOAD e ZOBEL, 2003). É necessário, no entanto, que ambos sejam originados a partir da mesma fonte. Por exemplo, um texto tem como co-derivadas as suas versões anteriores e subsequentes, versões adaptadas para outros formatos (página na *web*, formato de artigos ou periódicos) e qualquer documento produto de plágio a partir de tal texto (HOAD e ZOBEL, 2003). Destaca-se aqui a Wikipédia, cujos textos são produzidos a partir de revisões do mesmo artigo e que inclusive já foi utilizada para produzir *corpus* de co-derivação (BARRÓN-CEDEÑO, 2010, BARRÓN-CEDEÑO *et al.*, 2009).

Um observador humano consegue identificar se dois documentos são co-derivados através de algumas propriedades, a saber: palavras incomuns usadas em ambos os documentos; palavras que apresentem erros ortográficos; erros de pontuação; erros gramaticais; erros de citação copiados de uma fonte secundária e utilizações atípicas de determinadas palavras (HOAD e ZOBEL, 2003, MARTIN, 1994). Neste último caso, uma indicação de co-derivação, por exemplo, pode ser o uso incorreto de expressões como “mas” e “mais”, “onde” e “em que” e emprego inapropriado do verbo “haver”. Em trabalhos plagiados, é comum encontrar, além desse conjunto de características descritas, blocos idênticos de palavras, especialmente aqueles de tamanho extenso ou que contenham palavras não usuais (HOAD e ZOBEL, 2003).

2.2 Reúso Jornalístico

O problema em que a proposta deste trabalho será aplicada é o reúso de texto jornalístico. CLOUGH *et al.* (2002b) afirmam que o processo de edição e publicação de notícias em jornais é uma tarefa complexa e especializada, e, por diversas vezes, jornalistas são obrigados a recorrer às agências de notícias como fonte de suas matérias. Geralmente, os artigos preparados por jornais são condicionados a algumas restrições, tais como:

- Prazos curtos;
- Práticas prescritivas de escrita;

- Limitação de tamanho físico da matéria;
- Capacidade de leitura e compreensão do público-alvo;
- Viés editorial;
- Estilo do jornal.

Desta forma, optou-se por utilizar como base de publicações jornalísticas, o *corpus* METER. A criação desta base é detalhadamente descrita em (CLOUGH *et al.*, 2002b, GAIZAUSKAS *et al.*, 2001). Portanto, nesta seção, com base nas referências aqui citadas, faremos um breve resumo dos objetivos e características do *corpus*, além das classificações adotadas pelos autores para seu estudo e análise.

O Projeto METER (Measuring TExt Reuse) (CLOUGH *et al.*, 2002b), desenvolvido pelos departamentos de Jornalismo e Ciência da Computação da Universidade de Sheffield, foi responsável pela criação do *corpus* METER[4]. O projeto contou com a colaboração da Press Association[5] (PA), a maior agência de notícias do Reino Unido, que forneceu acesso ao seu serviço de notícias *online*.

Conforme explicado em (CLOUGH *et al.*, 2002b, GAIZAUSKAS *et al.*, 2001), a Press Association é uma grande fonte de notícias no Reino Unido, cujos textos são frequentemente reutilizados, direta ou indiretamente, nos jornais britânicos. Ainda que não usem diretamente a notícia, jornalistas, invariavelmente, referem-se ao serviço prestado pela Press Association durante a produção dos seus artigos. Portanto, em uma coleção composta por notícias produzidas por agências (como a PA) e textos de jornais de temas relacionados, esperam-se diferentes exemplos "reais" de reutilização de texto (GAIZAUSKAS *et al.*, 2001).

O objetivo da criação do projeto METER é a construção de um *corpus* capaz de oferecer material para o estudo e análise de reúso de texto e derivação em contexto jornalístico em que jornais utilizam matérias produzidas por agências de notícias. Os textos do *corpus* METER foram coletados manualmente a partir de um conjunto de notícias *online* escritas pela PA e dos textos correspondentes a estas notícias publicados em edições impressas de nove jornais britânicos (The Sun, Daily Mirror, Daily Star, Daily Mail, Daily Express, The Times, The Daily Telegraph, The Guardian e The Independent).

O *corpus* inclui publicações referentes à corte Britânica e ao *show business* (CLOUGH *et al.*, 2002b). Ambos os domínios de notícias foram escolhidos dada vasta quantidade de dados disponíveis e sua recorrência tanto nos jornais, quanto na PA. Com relação ao domínio corte Britânica, as notícias aqui incluídas abordam dados como o nome do acusado, a pena, e a localização de um julgamento ou

[4]http://nlp.shef.ac.uk/meter/
[5]http://www.pressassociation.com/

inquérito, o que deixa pouca margem para a interpretação jornalística. As notícias do segundo domínio do *corpus*, o *show business* (que também incluem temas de entretenimento), por outro lado, contrastam com as da corte Britânica. As publicações relacionadas ao *show business* são retratadas com maior liberdade de expressão e interpretação jornalística.

O *corpus* METER reúne uma coleção de 1.716 textos (mais de 500.000 palavras) coletados entre os meses de Julho de 1999 e Junho de 2000. O *corpus* está dividido em dois domínios, dos quais 1.429 são textos referentes a assuntos da corte Britânica e 287 são notícias do *show business*. Dentre os textos sobre a corte Britânica, 769 são artigos de jornais e 660 são notícias produzidas pela PA, distribuídos em 205 subcategorias. As notícias do *show business* se dividem em 175 artigos de jornais e 112 textos da PA, que foram alocadas em 60 subcategorias. Cada subcategoria se refere a uma história, incidente ou evento. É importante destacar que o foco principal do *corpus* METER são as notícias da corte Britânica, o que explica a menor quantidade de notícias do *show business* presentes na base.

O *corpus* foi classificado por um jornalista treinado de acordo com dois níveis de unidade de texto (nível de documento e nível de sequência de palavras, ou léxico), cada um deles com três graus de reúso. A descrição do processo de anotação e classificação encontra-se detalhada em (GAIZAUSKAS *et al.*, 2001). Como este trabalho tem por foco a utilização de técnicas para identificação de documentos co-derivados na base METER, explicaremos brevemente a classificação adotada pelos autores quanto à relação de dependência entre documentos da base. A saber, a classificação a nível de documento, define em que grau os textos publicações em jornais foram, como um todo, derivados de uma notícia da agência, ou seja, esta classificação busca representar a dependência do conteúdo de cada texto jornalístico em relação aos textos da PA. A classificação a nível de documento é dada pelas três categorias seguintes:

(a) **Completamente Derivado (CD):** Todo o conteúdo do texto-alvo (o artigo de jornal em questão) foi derivado somente da notícia publicada pela agência. Em um artigo de jornal desse grupo, todos os seus fatos podem ser mapeados para um ou mais textos da PA que pertencem à sua subcategoria. Os textos podem ser copiados na íntegra ou modificado de várias maneiras, incluindo reordenação de palavras, a substituição de sinônimos, e paráfrase.

(b) **Parcialmente Derivado (PD):** Parte do conteúdo do texto-alvo é derivado da notícia publicada pela agência, embora outras fontes também tenham sido utilizadas. Em publicações de jornal parcialmente derivadas, somente alguns trechos podem ser mapeados para o(s) texto(s) da PA correspondente(s). Esta categoria representa um grau intermediário de dependência entre textos de

jornal e publicações da PA.

(c) **Não Derivado (ND):** Nenhum conteúdo do texto-alvo é derivado da notícia publicada pela agência, mesmo que algumas palavras relacionadas possam ser comuns aos textos. Esta categoria abrange publicações de jornais que são escritas de forma independente dos artigos da PA. Ainda que tratando os mesmos temas que alguns textos da PA, os artigos dessa categoria não os utilizaram como fonte.

A nível léxico, palavras e frases de 400 documentos totalmente ou parcialmente derivados foram classificadas de acordo com a sua dependência em relação aos textos da agência de notícias. Novamente, três categorias foram usadas:

(a) **Verbatim:** o texto reutilizado foi integralmente copiado e utilizado no mesmo contexto;

(b) **Reescrito:** o texto reutilizado sofreu algumas alterações, isto é, foi parafraseado, passando a ter uma nova aparência, ainda que no mesmo contexto;

(c) **Novo:** o texto não aparece nas notícias da PA ou, se aparece, é usado em contexto diferente.

O *corpus* METER contém 300 artigos jornalísticos completamente derivados, 438 artigos parcialmente derivados e 206 artigos não derivados. Ou seja, no corpus, aproximadamente 78,2% dos documentos publicados em jornais são derivados das notícias escritas pela PA. Dentre as histórias coletadas, no domínio corte Britânica, 78,4% são consideradas derivadas (34% completamente derivadas e 44,4% parcialmente derivadas) e no domínio *show business*, 77,2% dos artigos são derivados (22,2% completamente derivados e 54,9% parcialmente derivados).

Capítulo 3

Heurísticas de Similaridade no Reúso de Texto

3.1 Introdução aos modelos adotados em tarefas de identificação de reúso

A tarefa de identificação de reúso é superficialmente similar à tarefa de busca ou classificação de documentos, estando estas relacionadas à semântica do documento e, em alguns casos, à estrutura sintática do mesmo (BERNSTEIN e ZOBEL, 2006). Um problema bastante recorrente durante a tarefa de identificação de reúso é encontrar, de forma eficiente, a fonte, através de um índice de milhões de documentos envolvendo milhares de palavras (ZHANG e CHOW, 2011).

Várias abordagens já foram propostas tentando resolver este problema. BAEZA-YATES *et al.* (1999) propõem, em seu livro, uma taxonomia para categorizar 15 modelos de Recuperação de Informação. Assim como todos os modelos matemáticos, os modelos de Recuperação de Informação fornecem um *framework* que possibilita a definição de novas tarefas e a compreensão dos resultados (CROFT *et al.*, 2010).

Contudo, três deles prevaleceram na literatura: Modelo Booleano, Modelo Vetorial e Modelo Probabilístico (APOSTOLICO *et al.*, 2006). No presente trabalho serão discutidas as métricas de similaridade utilizadas no modelo Booleano, no modelo Vetorial e em abordagens diferentes, como *Fingerprinting*, *Locality-Sensitive Hashing* e *Minwise Hashing*.

O modelo Booleano emprega um *framework* baseado na teoria dos conjuntos e na álgebra Booleana. As consultas são especificadas como expressões Booleanas com semântica precisa e somente são recuperados os documentos contendo termos que atendam à expressão lógica utilizada na consulta (BAEZA-YATES *et al.*, 1999, CARDOSO, 2000).

De acordo com BAEZA-YATES *et al.* (1999), seja uma consulta c, uma expressão

Booleana convencional; $\vec{c}_{fnd}$, a forma normal disjuntiva para a consulta c; e seja $\vec{c}_{cc}$ qualquer dos componentes conjuntivos de $\vec{c}_{fnd}$, a similaridade $sim_{bool}(d_i, c)$ de um documento d_i para a consulta c é definida por (3.1) (BAEZA-YATES *et al.*, 1999).

$$sim_{bool}(d_i, c) = \begin{cases} 1 & se\ \exists \vec{c}_{cc}\ \mid\ (\vec{c}_{cc} \in \vec{c}_{fnd}) \wedge (\forall_{t_j},\ \sigma_j(\vec{d_i}) = \sigma_j(\vec{c}_{cc})) \\ 0 & caso\ contrário \end{cases} \tag{3.1}$$

Onde t_j é um termo genérico e σ_j é uma função que retorna o peso associado ao termo t_j em qualquer vetor T-dimensional, sendo T é o número de termos de indexação.

Se a similaridade $sim_{bool}(d_i, c)$ for 1, o modelo prediz que o documento d_i é relevante para a consulta c. Caso contrário, o documento não será considerado relevante.

No modelo Vetorial, atribui-se valores não binários de pesos aos termos de indexação e aos termos da consulta com intuito de representar o grau de similaridade entre cada documento do *corpus* e a expressão de busca escolhida pelo usuário (BAEZA-YATES *et al.*, 1999, FERNEDA, 2003).

Neste modelo, tanto os documentos quanto a expressão de busca também são vistos como um vetor T-dimensional (CROFT *et al.*, 2010). Um documento d_i é representado pelo vetor $\vec{d_i} = (w_{i,1}, w_{i,2}, \ldots, w_{i,T})$, em que o peso $w_{i,j}$ associado ao par $[d_i, t_j]$ assume um valor positivo e não binário e t_j é um termo de indexação genérico. Analogamente, uma consulta c é representada pelo vetor $\vec{c} = (w_{c,1}, w_{c,2}, \ldots, w_{c,T})$.

Pode-se compreender a representação dos documentos e das consultas, no modelo Vetorial, como um vetor em um espaço T-dimensional, em que os T termos representam todas as caraterísticas presentes nos documentos que foram indexados (CROFT *et al.*, 2010).

Por meio desta representação, diferentes medidas de similaridade são utilizadas para quantificar a similaridade entre os dois vetores, sendo a medida de correlação do cosseno (sim_{cos}) a mais recorrente e consolidada (BAEZA-YATES *et al.*, 1999, CROFT *et al.*, 2010, MANNING *et al.*, 2008). De acordo com BAEZA-YATES *et al.* (1999), a quantificação da similaridade pode ser expressa pelo cosseno do ângulo entre dois vetores através da Equação (3.2).

$$sim_{cos}(d_i, c) = \frac{\vec{d_i} \cdot \vec{c}}{\mid \vec{d_i} \mid \times \mid \vec{c} \mid} = \frac{\sum_{j=1}^{T} w_{i,j} \times w_{c,j}}{\sqrt{\sum_{j=1}^{T} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{T} w_{c,j}^2}} \tag{3.2}$$

Onde o numerador desta medida é o produto escalar entre os pesos dos termos da consulta e do documento, enquanto $\mid \vec{d_i} \mid$ e $\mid \vec{c} \mid$ são as normas dos vetores d_i e c, respectivamente.

No que diz respeito à relação entre termos e documentos no modelo Booleano, as consultas são especificadas como expressões Booleanas com semântica precisa e somente são recuperados os documentos contendo termos que atendam à expressão lógica utilizada na consulta (BAEZA-YATES *et al.*, 1999, CARDOSO, 2000). Por causa da sua simplicidade e formalismo nítido, este modelo foi bastante adotado nos últimos anos (BAEZA-YATES *et al.*, 1999). Contudo, a incapacidade do modelo Booleano de conferir diferentes graus de relevância a cada documento apresenta-se como uma restrição ao seu uso e pode dificultar a obtenção de bons resultados com o uso desse modelo. Além dessa limitação, há ainda a complexidade que algumas operações Booleanas podem assumir, posto que as expressões empregadas na busca devem ser semanticamente precisas (BAEZA-YATES *et al.*, 1999). Expressões mais complexas requerem um sólido conhecimento em lógica Booleana por parte dos usuários (FERNEDA, 2003). Finalmente, outra dificuldade encontrada no uso do modelo Booleano é o pouco controle sobre a quantidade de documentos que são retornados em uma busca (FERNEDA, 2003). Isto pode levar o usuário a realizar diversas interações, procurando o refinamento lógico da expressão Booleana mais próximo do ideal, que retorne a quantidade desejada de documentos.

O modelo Vetorial reconhece as limitações do modelo Booleano ao empregar pesos binários aos termos de indexação em consultas e documentos (BAEZA-YATES *et al.*, 1999). De modo a evitar essas restrições, o modelo Vetorial, se diferencia do modelo Booleano através da forma como são calculados os pesos dos termos presentes no *corpus*. Diversos esquemas de atribuição de pesos foram utilizados ao longo dos últimos anos, entretanto, a maior parte deles são variações da medida *tf-idf* (CROFT *et al.*, 2010). Essa medida apresenta duas componentes, sendo uma delas a frequência dos termos (*term frequency*, ou *tf*) e a outra, o inverso da frequência do termo entre os documentos (*inverse document frequency*, ou *idf*).

BAEZA-YATES *et al.* (1999) fazem uma análise crítica do modelo Vetorial, citando as suas principais vantagens como: a estratégia de correspondência parcial, que permite a recuperação de documentos que atendam de forma aproximada à expressão de busca; o ranqueamento da similaridade por meio do cosseno do ângulo entre o vetor de busca e o vetor do documento e; a utilização do esquema de atribuição de pesos aos termos, que melhora o desempenho do modelo de recuperação.

Segundo a literatura, para o cálculo da similaridade em reúso de texto, existem outras técnicas desenvolvidas com o intuito de detectar co-derivados ou ranquear documentos que foram "copiados" da consulta (ZHANG e CHOW, 2011). Entre elas, este trabalho aborda a família de técnicas *Minwise Hashing*.

3.2 Algoritmos Aleatórios

De acordo com MOTWANI e RAGHAVAN (2010), um algoritmo aleatório (*randomized algorithm*, em inglês) é aquele que faz escolhas arbitrárias durante a sua execução. O algoritmo aleatório recebe, além dos seus dados de entrada, determinada quantidade de *bits* aleatórios que podem ser usados para o propósito de fazer escolhas aleatórias (KARP, 1991). Como um algoritmo aleatório utiliza números aleatórios para influenciar as escolhas que faz ao curso de sua computação, o seu comportamento varia de uma execução para outra, mesmo com uma entrada fixa (MOTWANI e RAGHAVAN, 1996).

O projeto e a análise de um algoritmo aleatório se baseiam na existência de um "bom" comportamento de cada entrada, dependente de escolhas probabilísticas feitas pelo algoritmo durante a execução e não em suposições sobre a entrada (MOTWANI e RAGHAVAN, 2010). MOTWANI e RAGHAVAN (1996) explicam que, ao se analisar um algoritmo aleatório, deve-se estabelecer limites para o valor esperado de uma medida de desempenho (por exemplo, o tempo de execução do algoritmo) válidos para cada entrada (MOTWANI e RAGHAVAN, 1996). A distribuição dessa medida do desempenho é posteriormente calculada sobre as escolhas arbitrárias feitas pelo algoritmo.

SLANEY e CASEY (2008) acrescentam ainda que o uso de algoritmos aleatórios não garante que sempre será obtida uma resposta exata, mas apresenta uma alta probabilidade de retornar essa resposta ou uma resposta próxima a ela. Entretanto, tal probabilidade pode ser aumentada com a adição de esforço e recursos computacionais.

Os algoritmos aleatórios vêm sendo utilizados de forma crescente, devido a dois grandes benefícios da randomização: a simplicidade e a rapidez (MOTWANI e RAGHAVAN, 1996). Muitas vezes os requisitos de espaço ou tempo de execução de um algoritmo aleatório são menores do que aqueles demandados pelos melhores algoritmos determinísticos conhecidos (KARP, 1991). Outra característica impressionante dos algoritmos aleatórios é que eles são extremamente simples de se entender e de aplicar; e muitas vezes, a introdução da randomização é suficiente para se converter um algoritmo determinístico com um "mau" comportamento de pior caso em um algoritmo aleatório com "boa" execução em cada possível entrada (KARP, 1991).

Alguns dos resultados provenientes de algoritmos aleatórios mais notáveis na área da Ciência da Computação, particularmente na teoria da complexidade, envolvem uma combinação não trivial de aleatoriedade e métodos algébricos (MOTWANI e RAGHAVAN, 2010). Essa combinação se faz presente nas técnicas de *Fingerprinting* e *Hashing*, frequentemente citadas na literatura no processo de verificação de

identidades envolvendo matrizes, polinômios e inteiros (MOTWANI e RAGHAVAN, 1996, 2010).

Ainda segundo MOTWANI e RAGHAVAN (1996), uma *fingerprint* é a "imagem de um elemento de um (grande) universo sob um mapeamento para outro universo (menor)". Os autores exemplificam, em seu trabalho, o uso das propriedades da *fingerprint* em problemas de correspondêcia de padrões, como o apresentado por KARP e RABIN (1987). Neste, é mostrado que duas cadeias de caracteres são propensas a serem idênticas se suas *fingerprints* forem iguais. Além disso, a comparação das *fingerprints* é consideravelmente mais rápida que a comparação das próprias cadeias de caracteres.

Em outro exemplo de técnica de randomização baseada em idéias algébricas, o *Hashing* (CARTER e WEGMAN, 1977), elementos de um conjunto C_i são armazenados em uma tabela de tamanho linear em $|C_i|$, com a garantia de que o número esperado de elementos em C_i mapeados para um determinado local na tabela é $O(1)$ (MOTWANI e RAGHAVAN, 1996). Portanto, ao usar o *Hashing*, a tarefa de descobrir se um elemento pertence ou não a C_i se torna muito mais eficiente.

É dado esse contexto que, nas seções subsequentes, será discutido o emprego de algoritmos aleatórios combinados com métodos algébricos. Serão abordadas as duas técnicas aqui apresentadas, *Fingerprinting* e *Hashing*. Entretanto, adotaremos uma visão mais direcionada ao problema de identificação de documentos similares.

3.3 *Fingerprinting*

De acordo com BARRÓN-CEDEÑO (2010), a técnica *Fingerprinting* vem sendo bastante utilizada para caracterizar e comparar documentos de acordo com o seu conteúdo. A abordagem *Fingerprinting*, voltada para encontrar documentos similares em grandes sistemas, teve origem em uma ferramenta desenvolvida por MANBER *et al.* (1994), conforme afirmam HEINTZE *et al.* (1996). A técnica era aplicada através da seleção de trechos de um documento e da subsequente utilização de funções *hash* para produzir as *fingerprints* (HEINTZE *et al.*, 1996).

A ideia por trás de uma *fingerprint* é fazer o mapeamento de uma *string* qualquer a partir de uma função matemática para um número inteiro aleatório, usando uma determinada chave (MANBER *et al.*, 1994). Ou seja, uma *fingerprint* $g(d)$ de um documento d, pode ser considerada um conjunto de *substrings* extraídas de d e posteriormente codificadas, que servem para identificar d de forma única (STEIN e ZU EISSEN, 2006). As *substrings* a serem mapeadas podem ser compostas de caracteres, palavras ou mesmo sentenças (BARRÓN-CEDEÑO, 2010).

As *fingerprints* funcionam como assinaturas de um documento. Portanto, uma característica dessa abordagem é que, dada uma alteração no documento, existe

uma alta probabilidade de que seja produzida uma *fingerprint* diferente (MANBER *et al.*, 1994). Através dessa técnica é possível fazer a comparação de *fingerprints* de diferentes documentos para determinar se eles são ou não produtos de reutilização textual (HOAD e ZOBEL, 2003, MANBER *et al.*, 1994).

Ao realizar a análise de um documento em comparação a uma coleção, gera-se uma *fingerprint* tanto para o documento pesquisado quanto para cada item pertencente à coleção. Se houver algum item da coleção cuja *fingerprint* seja compatível com aquela do documento pesquisado, deve-se fazer um *rank* com as pontuações por correspondência das representações das *substrings* que compõem as *fingerprints* (HEINTZE *et al.*, 1996, HOAD e ZOBEL, 2003). A complexidade do processo de comparação de um documento em relação a uma coleção é linear no número de documentos e no número de *substrings* da *fingerprint* (HOAD e ZOBEL, 2003).

HOAD e ZOBEL (2003), em seu trabalho sobre métodos para a identificação de documentos versionados e plagiados, enumeram e explicam as áreas compreendidas no processo de criação de *fingerprint*, sendo elas: geração, granularidade, tamanho e estratégia de seleção de *fingerprint*. Posteriormente, STEIN e ZU EISSEN (2006) sumarizaram e organizaram estas áreas da seguinte forma:

(a) Geração de *fingerprint*: Nesta etapa, as *substrings* selecionadas são mapeadas em números inteiros (STEIN e ZU EISSEN, 2006). O processo para a geração da *fingerprint* pode ter impacto significativo na eficácia do método (HOAD e ZOBEL, 2003). As funções de geração de *fingerprint* precisam satisfazer aos seguintes critérios: a geração deve poder ser reproduzida, os inteiros por ela gerados devem obedecer uma distribuição próxima à uniforme e a função deve ser rápida (HEINTZE *et al.*, 1996, HOAD e ZOBEL, 2003, MANBER *et al.*, 1994).

    A distribuição uniforme garante que a probabilidade de duas frases distintas representadas pelo mesmo inteiro seja, teoricamente, tão baixa quanto possível, entretanto, é inevitável que alguns pares de *strings* diferentes compartilhem a mesma representação (HOAD e ZOBEL, 2003). O algoritmo MD5 (*Message-Digest algorithm* 5) (RIVEST, 1992) é geralmente usado para a geração das *fingerprints* (STEIN e ZU EISSEN, 2006).

(b) Granularidade de *fingerprint*: O tamanho da *substring* extraída do documento determina a granularidade da *fingerprint* (HOAD e ZOBEL, 2003, STEIN e ZU EISSEN, 2006). A granularidade pode ser caracterizada de acordo com o grau de especificidade da *substring* analisada no documento textual, podendo ser a nível de número de caracteres, número de palavras ou de frases. Porém, BUTTLER (2004) indica que é mais fácil conceituá-la como o número de palavras na *string* (HOAD e ZOBEL, 2003).

A granularidade tem grande impacto na acurácia da abordagem *Fingerprinting* (HOAD e ZOBEL, 2003, SHIVAKUMAR e GARCIA-MOLINA, 1996). Quanto maior a granularidade, mais suscetível está a *fingerprint* a falsas correspondências entre documentos, enquanto que, quanto menor a granularidade, mais a *fingerprint* fica sensível a mudanças (STEIN e ZU EISSEN, 2006).

Ao usar *substrings* longas, um sistema somente é capaz de detectar trechos integralmente copiados ou casos de reúso em situações nas quais ambos os textos sejam idênticos (BARRÓN-CEDEÑO, 2010). Para identificar alguns casos de reúso com modificação, são mais indicadas representações menores, ainda que sob o risco de grande quantidade de falsos-positivos (BARRÓN-CEDEÑO, 2010, HEINTZE *et al.*, 1996).

(c) Tamanho da *fingerprint*: O tamanho, ou resolução, da *fingerprint* é o número inteiro usado para representar o documento (HOAD e ZOBEL, 2003). Conceitualmente, quanto mais informação é armazenada sobre um par de documentos, mais fácil é para decidir se eles são ou não reutilizados (HOAD e ZOBEL, 2003).

Entretanto, uma maior qualidade da *fingerprint* implica num aumento do esforço de processamento e nos requisitos de armazenamento, que devem ser cuidadosamente equilibrados (STEIN e ZU EISSEN, 2006).

(d) Estratégia de seleção de *fingerprint*: A partir dos documentos originais são extraídas *substrings* de acordo com alguma estratégia de seleção (STEIN e ZU EISSEN, 2006). A escolha de uma estratégia de seleção pode afetar tanto a acurácia quanto a eficiência do processo de geração de *fingerprint* (HEINTZE *et al.*, 1996, HOAD e ZOBEL, 2003).

A estratégia escolhida pode considerar informações referentes à posição, à frequência, ou informações estruturais no documento (STEIN e ZU EISSEN, 2006). Existem diversas estratégias, entretanto, este trabalho abordará somente aquelas identificadas por STAMATATOS (2011): *Fingerprinting* Completo (*Full Fingerprinting*) e Seletivo (*Selective Fingerprinting*).

3.3.1 *Fingerprinting* Completo

Esta estratégia de seleção consiste na escolha de sequências de caracteres de tamanho g_r no documento, onde g_r é a granularidade da *fingerprint* (HOAD e ZOBEL, 2003). Tal abordagem gera a maior resolução de *fingerprint* possível para o documento, e, por conseguinte, é também a estratégia mais custosa em termos de espaço de armazenamento (HEINTZE *et al.*, 1996, HOAD e ZOBEL, 2003).

Ao se utilizar o *Fingerprinting* Completo, a comparação entre dois documentos é feita através da contagem do número de *substrings* que são comuns a ambas as *fingerprints* (HEINTZE *et al.*, 1996). Posto que utiliza todas as *substrings* do documento, espera-se uma melhor eficácia ao usar o *Fingerprinting* Completo (HOAD e ZOBEL, 2003). Embora não seja prática, a abordagem é uma medida bastante útil para casos em que seja necessário medir a similaridade entre documentos (HEINTZE *et al.*, 1996).

Se escolhida apropriadamente, a granularidade g_r da *fingerprint* garante resultados confiáveis (HEINTZE *et al.*, 1996). Entretanto, como já visto anteriormente, a escolha de determinado valor de g_r pode levar a uma maior ou menor acurácia na busca por documentos reutilizados. Logo, fica claro que não existe um valor preciso de g_r para a abordagem que, de forma geral, possa ser medido quantitativamente ou empiricamente (HEINTZE *et al.*, 1996). Além disso, a escolha do valor de g_r é uma questão subjetiva, que depende do problema que se deseja analisar.

3.3.2 *Fingerprinting* Seletivo

O *Fingerprinting* Seletivo se diferencia do *Fingerprinting* Completo através da redução no tamanho da *fingerprint* utilizada. A redução se dá através da seleção de um subconjunto de *substrings* a partir da *fingerprint* completa para representar um documento (HEINTZE *et al.*, 1996).

O *Fingerprinting* Seletivo apresenta ainda duas divisões, sendo elas: tamanho fixo e tamanho proporcional (BARRÓN-CEDEÑO, 2010). Na primeira abordagem, escolhe-se um número fixo de *substrings*, independentemente do tamanho do documento (BARRÓN-CEDEÑO, 2010, HEINTZE *et al.*, 1996). Na segunda, o número de *substrings* selecionadas depende proporcionalmente do tamanho do documento (BARRÓN-CEDEÑO, 2010). O *Fingerprinting* Seletivo de tamanho proporcional, no entanto, tem a desvantagem de produzir *fingerprints* muito grandes de acordo com a quantidade de *substrings* que se pretende representar.

3.4 *Locality-Sensitive Hashing*

Problemas de identificação de itens semelhantes podem ser simplificados como uma busca pelo vizinho mais próximo a um objeto em dado espaço (SLANEY e CASEY, 2008). A solução do problema de encontrar vizinhos é relativamente simples, porém implica em varrer todos os itens de uma base de dados e ordená-los de acordo com a sua similaridade em relação ao objeto buscado (KULIS e GRAUMAN, 2009, SLANEY e CASEY, 2008).

Essa solução torna-se proibitiva quando a base de dados é muito extensa ou

quando a avaliação de similaridade entre pares de objetos é computacionalmente custosa (KULIS e GRAUMAN, 2009). Além disso, o tempo de processamento cresce linearmente com o número de itens e a complexidade dos objetos (SLANEY e CASEY, 2008). É nesse contexto que surge a técnica *Locality-Sensitive Hashing* (LSH), desenvolvida por INDYK e MOTWANI (1998), e posteriormente refinada por GIONIS *et al.* (1999).

O LSH foi criado para resolver problemas de alta dimensionalidade computacional, como a busca do vizinho mais próximo (BUHLER, 2001). Nessas pesquisas, o tempo computacional é drasticamente reduzido, com o custo de uma pequena probabilidade de não se conseguir encontrar o vizinho mais próximo ao item avaliado (SLANEY e CASEY, 2008). GIONIS *et al.* (1999) explicam que essa técnica é baseada na premissa de que, frequentemente, objetos semelhantes estão muito mais próximos do que aqueles que não são similares. Portanto, nessa situação, o algoritmo aproximado (com um fator de aproximação adequado) irá retornar o mesmo resultado que um algoritmo exato. Nos outros casos, o usuário terá que optar entre a qualidade dos resultados e o tempo necessário para a obtenção de uma resposta exata.

A ideia básica do LSH é gerar valores de *hashes* para diferentes objetos usando várias funções *hash*, de modo a assegurar que, para cada função, a probabilidade de colisão seja muito maior para objetos próximos do que para aqueles que estão distantes (GIONIS *et al.*, 1999). Ou seja, essa técnica aborda um simples conceito: se dois objetos são próximos, a sua projeção vetorial em um subespaço de menor dimensionalidade leva a dois pontos vizinhos (Figura 3.1) (SLANEY e CASEY, 2008). Assim, pode-se determinar os vizinhos mais próximos a um objeto calculando os *hashes* para o objeto e para os outros itens da base de dados e comparando os valores encontrados (GIONIS *et al.*, 1999).

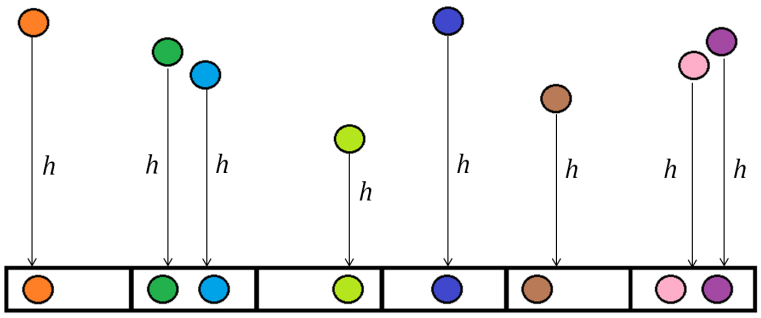

Figura 3.1: LSH: Objetos próximos são mapeados para o mesmo *slot*.
Adaptado de (HAUSBURG *et al.*, 2008).

O algoritmo usado pela técnica LSH é dividido em duas etapas, o pré-processamento e a consulta para encontrar os vizinhos mais próximos. Ambas as

etapas são explicadas mais a fundo em (GIONIS *et al.*, 1999). No presente trabalho apresentaremos somente os algoritmos de cada etapa, conforme detalhado pelos autores aqui citados.

O pré-processamento tem como entradas um conjunto C_i de J objetos e a variável N, que representa número de funções *hash* a serem usadas no LSH. As funções *hash* são responsáveis por fazer o mapeamento dos objetos para *slots* em tabelas *hash* (T_n; para $n = 1, \ldots, N$). Cada *slot* irá agrupar um conjunto de itens com propriedades semelhantes, isto é, o mesmo valor *hash*. As tabelas são as saídas do algoritmo de pré-processamento.

Algoritmo 1 PRIMEIRA ETAPA: PRÉ-PROCESSAMENTO

início

para cada $n = 1, \ldots, N$*:*

Inicialize a tabela hash T_n e gere uma função hash aleatória $h_n(\cdot)$

para cada $n = 1, \ldots, N$*:*

para cada $j = 1, \ldots, J$*:*

Mapeie o objeto o_j no *slot* $h_n(o_j)$ da tabela T_n

fim

retorna *Tabelas* $T_n; n = 1, \ldots, n$

Na fase de consulta dos vizinhos mais próximos, deve-se buscar os K itens que mais se aproximam do nosso objeto de busca c. É nesta etapa que são utilizadas as tabelas *hash* (T_n; $n = 1, \ldots, N$) criadas na fase de pré-processamento.

Algoritmo 2 SEGUNDA ETAPA: CONSULTA DOS VIZINHOS MAIS PRÓXIMOS

início

$C_K \leftarrow \varnothing$

para cada $n = 1, \ldots, N$*:*

$C_K \leftarrow C_K \cup \{$*objetos encontrados em* $h_n(c)$ *da tabela* $T_n\}$

fim

retorna K *vizinhos mais próximos de* c *encontrados no conjunto* C_K

O LSH deu origem a diferentes famílias de funções *hash*, sendo uma destas o *Minwise Hashing*. Ao longo da seção 3.5 explicaremos as características e particularidades desta família.

3.5 *Minwise Hashing*

O *Minwise Hashing* (BRODER, 1997, 2000, BRODER *et al.*, 1998) é uma instância particular da técnica *Locality-Sensitive Hashing* (CHARIKAR, 2002). A criação desse novo esquema foi motivada pelo aumento do conteúdo *online* disponível, acompanhado do crescimento na quantidade de páginas *web* e documentos *online* duplicados, de conteúdo reutilizado ou até plagiado, seja parcial ou integralmente (BRODER, 1997).

As distâncias normalmente utilizadas, como Hamming, Levenshtein, etc., não eram capazes de capturar essas características devidamente (BRODER, 1997, BRODER *et al.*, 1998). Não obstante demandarem um alto tempo de computação para a comparação entre documentos inteiros, a análise de extensos conjuntos de dados era impraticável, levando ao uso de mecanismos de amostragem por documento (BRODER, 1997). De modo a resolver este problema, BRODER (1997) desenvolveu o algoritmo *Minwise Hashing* (*min-Hash*). A ideia do algoritmo é descobrir a similaridade entre documentos por meio de um problema de interseção de conjuntos, que pode ser facilmente resolvido através de um processo de amostragem aleatória realizada de forma independente para cada documento (BRODER, 1997).

Ao representar documentos como conjuntos de características e utilizar o índice de Jaccard (JACCARD, 1908) como métrica de similaridade entre esses conjuntos, o *min-Hash* proporciona um método simples para estimar a similaridade de documentos, permitindo que os documentos originais sejam descartados e reduzindo significativamente o tamanho da entrada (CHARIKAR, 2002).

O índice de Jaccard, ou coeficiente de similaridade de Jaccard, é uma medida estatística usada para calcular a similaridade entre conjuntos dada a razão dos tamanhos relativos da sua união e da sua interseção (RAJARAMAN e ULLMAN, 2011). Essa medida é utilizada por BRODER *et al.* em seus trabalhos (BRODER, 1997, 2000, BRODER *et al.*, 1998) para calcular a similaridade entre dois conjuntos C_1 e C_2. O índice de Jaccard ($sim_{jac}(A, B)$) entre C_1 e C_2 é definido pela Equação (3.3) (BRODER, 2000, BRODER *et al.*, 1998).

$$sim_{jac}(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} \tag{3.3}$$

A similaridade de Jaccard se destina a encontrar documentos textuais em grandes coleções de dados, como a *web* ou *corpora* de artigos de notícias (RAJARAMAN e ULLMAN, 2011). Testar se dois documentos são cópias exatas é uma tarefa fácil: basta comparar os dois documentos caractere a caractere, e se houver alguma diferença, pode-se concluir que eles não são os mesmos (RAJARAMAN e ULLMAN, 2011).

No entanto, em muitas aplicações, os documentos não são idênticos, mas par-

tilham grandes porções dos seus textos (RAJARAMAN e ULLMAN, 2011). Por esse motivo, como o objetivo do algoritmo *min-Hash* é encontrar todos os documentos na base de dados que sejam similares a um documento pesquisado, estes serão considerados similares se a similaridade entre eles for maior que um dado *threshold* (CHUM *et al.*, 2008), como será visto na seção 3.5.1.

3.5.1 *Shingling*

Com o intuito de associar cada documento a um conjunto de subsequências de *tokens*, BRODER (1997) desenvolveu uma técnica chamada *Shingling* capaz de tornar possível a comparação de dois documentos textuais. Esta técnica reduz conjuntos de palavras, ou *tokens*, em um documento a uma lista de *hashes* que podem ser comparados diretamente com outro documento usando diferença, união e interseção de conjuntos para determinar a similaridade (BUTTLER, 2004).

O modo mais eficiente de representar documentos como conjuntos, para o propósito de identificar aqueles que são lexicalmente semelhantes, é elaborar um conjunto de pequenos *tokens* que aparecem em cada documento, sejam eles caracteres, palavras ou sentenças (BRODER, 1997, RAJARAMAN e ULLMAN, 2011). Assim, dois documentos que compartilhem determinados trechos, não necessariamente na mesma ordem, terão elementos em comum em seus conjuntos.

De acordo com BRODER (1997), uma sequência adjacente contida em um documento d é chamada de *shingle* e a técnica tem por objetivo associar cada documento a um conjunto de subsequências de *tokens*. Mais especificamente, dado um documento d, o seu *w-shingling* é definido como um conjunto de todos os *shingles* de tamanho w contidos em d (BRODER, 1997).

Por exemplo, para um documento $d = (abcdbcd)$, e $w = 3$, o conjunto *3-shingles* de d será $\{abc, bcd, cdb, dbc\}$.

Em vez de lidar com as *shingles* diretamente, BRODER (2000) calcula a *fingerprint* de cada *shingle*. Isto é feito sob a justificativa de que as *fingerprints* produzidas são relativamente curtas quando se considera grandes objetos. Além disso, uma propriedade bastante interessante é que se duas *fingerprints* forem distintas, então os seus respectivos objetos também serão diferentes, existindo apenas uma pequena chance de que dois objetos distintos tenham a mesma *fingerprint*. Essa probabilidade é exponencialmente pequena no comprimento da *fingerprint*.

Por questões de eficiência computacional, aqueles *tokens* selecionados para compor os *shingles* geralmente são codificados através de uma função *hash* (BARRÓN-CEDEÑO, 2010, STAMATATOS, 2011). Subconjuntos de *shingles*, chamados "esboços" (*sketchs*), são então usados para calcular a similaridade entre documentos (BUTTLER, 2004). Os "esboços" são uma amostra aleatória de texto de um docu-

mento e podem ser rapidamente gerados (complexidade de tempo linear no tamanho dos documentos) (BRODER, 2000, BUTTLER, 2004). Basicamente, a comparação de documentos é feita por meio da comparação de alguns dos seus "esboços" selecionados de forma aleatória. Considera-se, nesta comparação, que a sobreposição entre diferentes amostras indica uma forte tendência a uma sobreposição também entre os documentos. O uso desta técnica possibilita, dados dois "esboços", computar o valor da similaridade entre os documentos correspondentes em tempo linear no tamanho dos "esboços" (BRODER, 1997, 2000). Adicionalmente, uma comparação entre documentos de qualquer tamanho pode ser feita em tempo constante, tendo como penalidade uma leve redução da acurácia, como pôde ser mostrado no trabalho de BUTTLER (2004).

Se o *shingle* avaliado for grande o suficiente (sentenças, parágrafos ou até o documento todo, por exemplo), ele pode indicar uma propensão à correspondência entre documentos distintos (BARRÓN-CEDEÑO, 2010). Caso contrário, ainda é possível calcular a similaridade entre eles usando um valor de *threshold* (BARRÓN-CEDEÑO, 2010). Valores altos de *threshold* são mais indicados para a busca de documentos com altos índices de reúso (grandes porções de texto copiadas), enquanto que valores mais baixos de *threshold* são mais apropriados para detectar pequenos trechos reaproveitados (SHIVAKUMAR e GARCIA-MOLINA, 1996).

É importante ressaltar que alguns autores, como BERNSTEIN e ZOBEL (2004), LYON *et al.* (2001) e BRIN *et al.* (1995), também chamam as sequências adjacentes compostas por palavras ou caracteres em um documento de *chunks*. FORMAN *et al.* (2005), no entanto, consideram que a técnica *Chunking* consiste em particionar um documento em pequenos trechos, como parágrafos; enquanto que um *shingle* corresponde a uma *fingerprint* calculada sobre cada posição de uma janela de tamanho fixo que desliza pelo documento.

3.5.2 Definição do Problema da Interseção de Conjuntos e Estimativa da Similaridade

No algoritmo *min-Hash*, BRODER (1997) emprega uma permutação, definida como π, dos grupos de objetos pertencentes a cada conjunto C_i do universo C por meio de uma função *hash* h. A partir de dada permutação, o valor da assinatura mínima (*min-Hash*) de cada conjunto C_i será igual ao primeiro menor elemento pertencente a C_i, considerando-se a ordem permutada induzida por h (CHUM *et al.*, 2008).

CHUM *et al.* (2008) simplificam a explicação do algoritmo desenvolvido por BRODER (1997) da seguinte forma: Sejam t_1 e t_2 dois termos diferentes extraídos do vocabulário de C, as funções *hash* devem satisfazer a duas condições $h(t_1) \neq h(t_2)$

e $P(h(t_1) < h(t_2)) = 0,5$. Além disso, as funções h devem também ser independentes (CHUM *et al.*, 2008). Matematicamente, pode-se expressar o *min-Hash* (*min*) de um conjunto C_i através da Equação (3.4) (CHUM *et al.*, 2008).

$$min(C_i, h) = \arg\min_{t \in C_i} h(t) \tag{3.4}$$

O método *min-Hash* é baseado na probabilidade de que $min(C_1, h) = min(C_2, h)$ seja dado por (3.5) (BRODER, 2000, CHUM *et al.*, 2008).

$$\mathbf{Pr}(min(C_1, h) = min(C_2, h)) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} = sim_{jac}(C_1, C_2) \tag{3.5}$$

Ao estimar $sim_{jac}(C_1, C_2)$, são utilizadas N funções *hash* h independentes (CHUM *et al.*, 2008). Considerando a quantidade de vezes que $min(C_1, h) = min(C_2, h)$ igual a l, podemos constatar que l segue a distribuição binomial $Bi(N, sim_{jac}(C_1, C_2))$ e que a estimativa de probabilidade máxima de $sim_{jac}(C_1, C_2)$ é l/N (CHUM *et al.*, 2008).

Fazendo o objeto $o_j = min(C_1 \cup C_2, h)$, como h é uma função *hash* aleatória, cada elemento de $C_1 \cup C_2$ tem a mesma probabilidade de ser o menor (BRODER, 2000, CHUM *et al.*, 2008). Desta forma, podemos retirar o_j de forma aleatória de $C_1 \cup C_2$. Se o_j pertencer tanto a C_1 quanto a C_2 ($C_1 \cap C_2$), teremos $min(C_1, h) = min(C_2, h) = o_j$ (BRODER, 2000, CHUM *et al.*, 2008). Caso contrário, se o_j pertencer somente a C_1 ou somente a C_2, teremos $o_j = min(C_1, h) \neq min(C_2, h)$ ou $min(C_1, h) \neq min(C_2, h) = o_j$, respectivamente (CHUM *et al.*, 2008). Portanto, a Equação (3.5) define que o_j é retirado de $|C_1 \cup C_2|$ de modo aleatório e que a igualdade de *min-Hashes* ocorre em $|C_1 \cap C_2|$ casos (CHUM *et al.*, 2008).

Assim sendo, podemos escolher um conjunto de N permutações aleatórias induzidas pelas funções *hashes* $\{h_1, h_2, \cdots, h_N\}$, gerando, para cada documento, um "esboço" dado por (3.6) (BRODER, 2000). Logo, a similaridade entre C_1 e C_2 pode ser estimada ao contabilizarmos quantos elementos correspondentes em $\overline{C}_{min,1}$ e $\overline{C}_{min,2}$ são iguais.

$$\overline{C}_{min,i} = (min(C_i, h_1), min(C_i, h_2), \cdots, min(C_i, h_N)) \tag{3.6}$$

3.5.3 Representação Matricial

Para fins didáticos, antes de iniciar a construção das assinaturas dos documentos, deve-se observar a interação entre conjuntos e elementos através de uma representação matricial, a matriz característica M, na qual as colunas de M correspondem aos conjuntos e as linhas, a cada elemento do conjunto universo C_i. A posição $M(j, i)$ de M é dada por 1 se o elemento j estiver presente no conjunto i. Caso contrário,

$M(j, i)$ será 0.

Exemplo: Sejam os conjuntos C_1, C_2 e C_3 dados por $\{a, c, e\}$, $\{b, c, d\}$ e $\{b, e\}$, respectivamente, em um universo igual a $\{a, b, c, d, e\}$. Logo, a representação dos conjuntos na forma matricial é dada pela Tabela 3.1.

	C_1	C_2	C_3
a	1	0	0
b	0	1	1
c	1	1	0
d	0	1	0
e	1	0	1

Tabela 3.1: Exemplo de construção da representação matricial dos conjuntos C_1, C_2 e C_3.

3.5.4 Implementação do Algoritmo *min-Hash*

Usando a representação matricial, obtém-se a assinatura *min-Hash* de um conjunto C_i dado por uma coluna da matriz característica M por meio da escolha de uma permutação aleatória das linhas da matriz. O valor *min-Hash* de qualquer coluna referente à C_i será o número da primeira linha que tenha um 1, na ordem permutada (RAJARAMAN e ULLMAN, 2011).

As assinaturas para cada conjunto são construídas através dos resultados de um grande número de cálculos, sendo cada um dos quais um *min-Hash* das colunas da matriz característica M (RAJARAMAN e ULLMAN, 2011). Para exemplificar uma permutação da matriz característica M usada no exemplo anterior (Tabela 3.1), escolhe-se a seguinte permutação aleatória de linhas: *daceb*. A matriz correspondente a essa permutação é dada pela Tabela 3.2.

	C_1	C_2	C_3
d	0	1	0
a	1	0	0
c	1	1	0
e	1	0	1
b	0	1	1

Tabela 3.2: Permutação das linhas da matriz característica M dos conjuntos C_1, C_2 e C_3.

RAJARAMAN e ULLMAN (2011) afirmam que a operação de permutar uma matriz característica muito extensa é computacionalmente custosa, além de demandar um alto tempo de processamento. Os autores também explicam que a ineficiência computacional das permutações de matrizes acaba dificultando o seu uso, ainda que este seja conceitualmente interessante. Porém, felizmente, é possível simular o efeito

de uma permutação aleatória através de funções *hash* aleatórias que realizam o mapeamento dos índices das linhas para o mesmo número de *slots* (RAJARAMAN e ULLMAN, 2011).

Ainda que uma função *hash* possa mapear diferentes inteiros correspondentes aos índices das linhas da matriz M para o mesmo *slot* enquanto outros *slots* ficam vazios, as colisões não são relevantes (RAJARAMAN e ULLMAN, 2011). Este fato se explica porque o número de índices é bastante alto, o que reduz consideravelmente a quantidade de colisões. A distribuição dos índices em diferentes *slots* funciona, desta forma, como uma permutação, levando a alocação da linha L em uma posição *h(L)* na ordem permutada pela função *hash* h (RAJARAMAN e ULLMAN, 2011).

Assim, cada permutação π_n escolhida define uma função *hash* h_n que mapeia a matriz característica para outra matriz permutada (RAJARAMAN e ULLMAN, 2011). Neste caso, o valor da assinatura da função *hash* h_n para um conjunto C_i é o elemento da linha correspondente à primeira ocorrência de um 1 na coluna referente à C_i. Portanto, ao computar o valor do *min-Hash* para os conjuntos C_1, C_2 e C_3 de acordo com h_n, tem-se $min(C_1, h_n) = a$, $min(C_2, h_n) = d$ e $min(C_3, h_n) = e$, respectivamente.

Ao representar conjuntos através de funções *hash*, é necessário escolher certa quantidade de permutações aleatórias. Seja N o número de permutações, esse valor pode variar de acordo com as variáveis do problema (no caso dos exemplos, números de elementos do universo). Assim sendo, para a construção da matriz de assinatura S, em vez de escolher N permutações aleatórias de linhas, escolhe-se N funções *hash* aleatórias $\{h_1, h_2, \cdots, h_N\}$ (RAJARAMAN e ULLMAN, 2011).

A partir do conjunto de funções *hash* aleatórias e da matriz característica M, pode-se construir uma matriz de assinatura S, cuja i-ésima coluna corresponde à assinatura *min-Hash* para a i-ésima coluna de M. As colunas S_i da matriz de assinatura S serão dadas pelos vetores formados pelas assinaturas *min-Hash* de C_i, sendo $S_i = [min(C_i, h_1), min(C_i, h_2), \cdots, min(C_i, h_N)]$. Esses vetores, na verdade, correspondem aos "esboços" (3.6) e é através deles que se pode comparar dois diferentes conjuntos.

Como exemplo da construção de uma matriz de assinaturas S, tendo $N = 3$, escolhemos as seguintes funções *hash* h_1, h_2, h_3 que induzem às permutações $abcde, daceb, eabcd$, respectivamente. A matriz S produzida a partir dessas permutações é dada pela Tabela 3.3.

Os valores das assinaturas da matriz S são armazenados em *slots*. Portanto, para todo conjunto C_i pertencente a M e toda função *hash* h_n, um $slot(h_n, C_i)$ é preparado para guardar a sua assinatura *min-Hash* segundo a função *hash* h_n.

Note ainda que a matriz de assinatura S tem o mesmo número de colunas de M, mas apenas N linhas (RAJARAMAN e ULLMAN, 2011). Mesmo que M não es-

	S_1	S_2	S_3
h_1	a	b	b
h_2	a	d	e
h_3	e	b	e

Tabela 3.3: Matriz assinatura dos conjuntos C_1, C_2 e C_3 dadas as permutações produzidas pelas funções *hash* h_1, h_2, h_3.

teja representada explicitamente, mas de alguma forma comprimida (representação esparsa), é comum que a matriz de assinatura seja muito menor que M (RAJARAMAN e ULLMAN, 2011).

O cálculo do *min-Hash* para um conjunto C_i pertencente a M é dado pelo seguinte algoritmo (PHILLIPS, 2012, RAJARAMAN e ULLMAN, 2011):

Algoritmo 3 Cálculo da assinatura *min-Hash* do conjunto $C_i \in M$

início

Inicialize todo $slot(h_n, C_i)$ com ∞

para cada *permutação dada por* h_n*, com* n *variando de 1 até* N*:*

para cada *linha* L_j *de* M*:*

se $M(L_j, C_i)$ *for igual a 1* **então**

se $h_n(j) < slot(h_n, C_i)$ **então** — Seleção da assinatura

$slot(h_n, C_i) \leftarrow h_n(j)$

fim

retorna *Assinatura min-Hash do conjunto* C_i

Finalmente, deve-se calcular a similaridade par a par (Equação (3.7)) (PHILLIPS, 2012) entre as assinaturas dos conjuntos.

$$sim_{jac}(\overline{C}_{min,1}, \overline{C}_{min,2}) = \frac{1}{N} \sum_{n=1}^{N} \gamma(\overline{C}_{min,1}(n), \overline{C}_{min,2}(n)) \tag{3.7}$$

Onde $\overline{C}_{min,i}(n)$ é a assinatura *min-Hash* do conjunto C_i segundo a função *hash* h_n e

$$\gamma(A, B) = \begin{cases} 1 & se\ A = B \\ 0 & caso\ contrário \end{cases} \tag{3.8}$$

Capítulo 4

Métodos Propostos

4.1 Semi-Reticulados e Reticulados

Considerando A um conjunto parcialmente ordenado não-vazio, diz-se que $a \in A$ é um limite superior de um subconjunto $B \subseteq A$, ou que a limita superiormente B quando, para todo $b \in B$, $b \leq a$ (PRATT, 2015). O menor limite superior de B, chamado supremo, é aquele menor ou igual a todo limite superior de B (PRATT, 2015, ROMAN, 2008). Analogamente, o limite inferior de B é a se $a \leq b$ para todo $b \in B$ (PRATT, 2015). Logo, o maior limite inferior de B é chamado ínfimo, sendo este maior ou igual a todo limite inferior de B (ROMAN, 2008).

Semi-reticulados (*semilattices*, em inglês) podem ser definidos de duas maneiras (ROMAN, 2008). A primeira é baseada na existência de uma relação de ordem que obedece a certas propriedades e a segunda é baseada na existência de operações binárias que satisfaçam determinadas propriedades algébricas (ROMAN, 2008).

A definição de semi-reticulados baseada em ordem é a de que um semi-reticulado é um conjunto parcialmente ordenado, fechado em uma de duas operações binárias, ou o supremo ou o ínfimo (NEZHAD e DAVVAZ, 2009). Logo, sendo S um conjunto parcialmente ordenado pela relação binária $\leq$, $\mathbf{S} = (S, \leq)$ é um semi-reticulado inferior se, cada par $(a, b) \in S$ existir um limite inferior máximo, denotado por $a \wedge b$ (NEZHAD e DAVVAZ, 2009). Por outro lado, o limite superior mínimo de cada par (a, b) resulta em um conceito análogo, o semi-reticulado superior, denotado por $a \vee b$ (NEZHAD e DAVVAZ, 2009).

Segundo ROMAN (2008), em um semi-reticulado inferior pode existir um único elemento mínimo ou não haver mínimo algum. O mesmo é válido para o semi-reticulado superior, conservadas as devidas propriedades.

Também pode-se definir um semi-reticulado $\mathbf{S} = (S, *)$ como um par constituído por um conjunto S e uma operação binária $*$, que apresenta as seguintes propriedades: comutatividade, associatividade e idempotência (NATION, 2009). Assim, para

todo $a, b, c \in S$, segundo NATION (2009), tem-se :

(a) Idempotência: $a * a = a$,

(b) Comutatividade: $a * b = b * a$,

(c) Associatividade: $a * (b * c) = (a * b) * c$.

O símbolo $*$ pode ser substituído por qualquer símbolo de operação binária, como $\vee, \wedge, +$, ou $\cdot$, dependendo da configuração utilizada (NATION, 2009).

Um reticulado (*lattice*, em inglês) é definido como um conjunto parcialmente ordenado que é simultaneamente um semi-reticulado superior e um semi-reticulado inferior (PRATT, 2015).

Seja R um conjunto parcialmente ordenado, $\mathbf{R} = (R, \wedge, \vee)$ é um reticulado se todo par de elementos de R tiver um limite superior mínimo e um limite inferior máximo (ROMAN, 2008). Um reticulado conserva as propriedades dos semi-reticulados: comutatividade, associatividade e idempotência. A ligação entre as duas operações $\vee$ e $\wedge$ é fornecida por meio das leis de absorção (Equação 4.1) (ROMAN, 2008).

$$a \wedge (a \vee b) = a \; e \; a \vee (a \wedge b) = a \tag{4.1}$$

Ou seja, um reticulado é caracterizado pelas seguintes propriedades, enumeradas por BEDREGAL *et al.* (2006), ROMAN (2008), para todo $a, b, c \in R$:

(a) Idempotência: $a \wedge a = a$ e $a \vee a = a$

(b) Comutatividade: $a \wedge b = b \wedge a$ e $a \vee b = b \vee a$

(c) Associatividade: $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ e $a \vee (b \vee c) = (a \vee b) \vee c$

(d) Absorção: $a \wedge (a \vee b) = a$ e $a \vee (a \wedge b) = a$

Caso R seja um conjunto não-vazio com dois operadores binários $\vee$ e $\wedge$, satisfazendo às propriedades citadas, então $\mathbf{R}$ é um reticulado onde o limite superior mínimo é $\wedge$, o limite inferior máximo é $\vee$ e a ordem da relação é dada pela Equação (4.2), onde $\wedge$ coincide com o maior limite inferior (ínfimo) e $\vee$ coincide com o menor limite superior (supremo) (ROMAN, 2008).

$$\begin{aligned} a \leq b \; se \; a \vee b = b \\ a \leq b \; se \; a \wedge b = a \end{aligned} \tag{4.2}$$

Após observar os conceitos de semi-reticulado e reticulado, pode-se concluir que o método *Minwise Hashing*, na verdade um semi-reticulado inferior, busca descobrir

qual é o limite inferior máximo de um conjunto, ou seja, seu ínfimo, para poder compará-lo com outro. Isso traz imensas vantagens ao *min-Hash*, que é capaz de gerar um retrato parcial dos conjuntos selecionados. Desta forma, não é preciso percorrer os elementos de dois conjuntos par a par e comparar a sua interseção.

Porém, do mesmo modo que exploramos o ínfimo de um conjunto, podemos estudar o seu supremo através da descoberta do maior elemento nele presente. Logo, por meio de um semi-reticulado superior, também descobrimos a fronteira superior do conjunto, e, assim criamos um novo método, que busca utilizar essa fronteira para gerar outro retrato parcial dos conjuntos. Esse novo método, que recebe o nome de *Maxwise Hashing*, será melhor abordado na seção 4.3.1.

Outra abordagem que tratamos no presente estudo consiste em analisar simultaneamente tanto os limites inferiores quanto os superiores dos conjuntos. Neste caso, busca-se explorar um dado conjunto sob a ótica de um reticulado, considerando as duas fronteiras do mesmo (seu ínfimo e seu supremo). A este método daremos o nome de *MinMaxwise Hashing*. Na seção 4.3.3 demonstraremos o seu desenvolvimento.

Também nos dedicaremos ao estudo da utilização de dois mínimos ou dois máximos para representar um dado conjunto. A essas abordagens, chamaremos de *MinMixwise Hashing* e *MaxMaxwise Hashing*, respectivamente. Ambas serão desenvolvidas na seção 4.3.2.

4.2 Operadores de Agregação

De acordo com PEDRYCZ e GOMIDE (1998), os elementos de uma coleção de conjuntos *fuzzy* podem ser combinados de modo a produzir um único conjunto *fuzzy* através de operações de agregação. Desta forma, a representação de um conjunto *fuzzy* através do seu mínimo, como realizado no algoritmo *Minwise Hashing*, pode ser considerada uma operação de agregação.

PEDRYCZ e GOMIDE (1998) definem uma operação de agregação como uma operação n-ária $A : [0,1]^n \to [0,1]$ que satisfaz os seguintes requisitos:

(a) Condição de fronteira: $A(0,\ldots,0) = 0$ e $A(1,\ldots,1) = 1$

(b) Monotonicidade: $A(x_1 \ldots, x_n) \geq A(y_1,\ldots,y_n)$ se $x_i \geq y_i, i = 1,\ldots,n$.

Operadores de agregação OWA (*Ordered Weighted Average*) (YAGER, 1988) modelam um processo de agregação no qual uma sequência A de n valores escalares é ordenada de modo decrescente e tem pesos atribuídos aos seus elementos de acordo com posição que ocupam através de um vetor de pesos $w = (w_1, w_2, \ldots, w_n)$, onde

cada elemento $w_i \in w$ está limitado pelo intervalo $[0, 1]$ e $\sum_{i=1}^{n} w_i = 1$ (CORNELIS *et al.*, 2010).

Seja a sequência do conjunto de valores $A(x_i)$ ordenada como: $A(x_1) \leq A(x_2) \leq \cdots \leq A(x_n)$, então $OWA(A, w)$ é dada pela Equação (4.3) (PEDRYCZ e GOMIDE, 1998).

$$OWA(A, w) = \sum_{i=1}^{n} w_i A(x_i) \tag{4.3}$$

Segundo FODOR *et al.* (1995) e CORNELIS *et al.* (2010), a classe de operadores OWA inclui diferentes operações, dentre as quais citamos:

- mínimo$(w_1 \ldots, w_n)$, se $w_n = 1, w_i = 0$, para $i \neq n$
- máximo$(w_1 \ldots, w_n)$, se $w_1 = 1, w_i = 0$, para $i \neq n$
- a média aritmética, se $w_i = 1/n$, para $i = 1, \ldots, n$
- qualquer ordem estatística x_i se $w_i = 1, i = 1, \ldots, n$

FODOR *et al.* (1995) acrescentam ainda que qualquer operador de agregação OWA tem como propriedades a neutralidade, monotonicidade e idempotência, além de apresentar estabilidade para as mesmas transformações positivas lineares. O operador OWA, entretanto, não é associativo ou passível de decomposição.

Observando as operações OWA enumeradas, é possível constatar que o mínimo leva a uma representação *fuzzy* de agregação entre conjuntos. Da mesma forma, outros operadores também podem ser estudados, como o máximo, no caso do *Maxwise Hashing*.

Há ainda a possibilidade de se explorar duas operações de agregação para representar uma relação entre conjuntos *fuzzy*, como, por exemplo, estudar simultaneamente o máximo e o mínimo de um conjunto (*MinMaxwise Hashing*), os seus dois mínimos (*MinMixwise Hashing*) ou seus dois máximos (*MaxMaxwise Hashing*).

4.3 Variantes do método *Minwise Hashing*

Pode-se compreender o *slot* utilizado no conjunto de técnicas LSH (seção 3.4) e, consequentemente, na família de funções *Minwise Hashing* (seção 3.5.4), como o resumo de um conjunto que ele pretende representar conforme um *operador*. O *slot* é construído considerando-se dois parâmetros: um conjunto de índices e o *operador* utilizado na sua criação.

Operadores são diferentes conjuntos de características que podem sumarizar a representação de documentos. No caso do *Minwise Hashing*, o *operador* explorado é

o mínimo de um conjunto. Porém, esta é uma escolha arbitrária dentre um conjunto de possíveis *operadores* que podem ser estudados, como o mínimo, o máximo, o i-ésimo mínimo e o i-ésimo máximo de um conjunto. Portanto, supondo O como o universo de todos os *operadores*, a Tabela 4.1 enumera exemplos de subconjuntos de *operadores* ($O_k \subset O$) que podem ser usados na representação de versões resumidas de documentos.

k	O_k
min	$\{min_1\}$
max	$\{max_1\}$
$minMax$	$\{min_1, max_1\}$
$minMin$	$\{min_1, min_2\}$
$maxMax$	$\{max_1, max_2\}$
$média$	$\{média\}$
$minMinMax$	$\{min_1, min_2, max_1\}$
$min \cdots min$	$\{min_1, \cdots min_N\}$

Tabela 4.1: Exemplos de $O_k \subset O$.

Neste trabalho, empregaremos diferentes formas de representar o mesmo conjunto. Além do mínimo sugerido pelo algoritmo *Minwise Hashing*, também estudaremos outros tipos de *operadores*, como o seu máximo. Outra forma de observar os conjuntos é por meio da exploração de uma combinação de *operadores*. Nesse caso, podemos analisar um conjunto ao olhar para o seu máximo e seu mínimo, simultaneamente, ou ainda para os seus dois menores ou dois maiores elementos.

Ao observar um conjunto sob duas óticas diferentes, utilizaremos dois tipos de *slots* e, portanto, teremos duas vezes mais informações do conjunto do que teríamos usando um *operador* simples (*min, max, média*). Analisando um conjunto a partir de uma quantidade fixa de permutações, e estudando, por exemplo, o mínimo e o máximo produzidos por cada permutação para os conjuntos, teremos os seguintes *slots*: $slot_{min}$ e $slot_{max}$. Ou seja, nesse caso, serão gerados $2(N \times m)$ *slots*, onde N é o número de permutações e m é a quantidade de conjuntos (documentos) do *corpus*.

4.3.1 *Maxwise Hashing*

Uma das ideias propostas nesse trabalho consiste em, além de olhar para o menor elemento dos conjuntos, considerar também o seu máximo. Ao utilizar a maior assinatura gerada para cada conjunto analisado, espera-se obter resultados semelhantes aos alcançados com o uso do *min-Hash*. Assim como o mínimo equivale ao maior limite inferior de um conjunto, de modo análogo, o máximo irá corresponder ao seu menor limite superior. Ou seja, tomando como referência o método *Minwise Hashing*, que explora o ínfimo de um conjunto, podemos também extrair o supremo do mesmo, e teremos, portanto, o seu semi-reticulado superior.

Motivados por essa ideia, desenvolvemos um método chamado *Maxwise Hashing*. Nele, definimos a assinatura máxima (*max-Hash*) de cada conjunto C_i como o primeiro maior elemento pertencente a C_i, de acordo com a ordem permutada aleatoriamente induzida por uma função *hash* h. Logo, assim como a assinatura *min-Hash* é dada pela Equação (3.4), a assinatura *max-Hash* pode ser obtida através da Equação (4.4).

$$max(C_i, h) = \arg\max_{t \in C_i} h(t_j) \tag{4.4}$$

Ao considerar a permutação aleatória dada por h, e calcular a assinatura máxima da união de dois conjuntos C_1 e C_2, podemos deduzir que cada elemento de $C_1 \cup C_2$ terá também a mesma probabilidade de ser o maior elemento da união. Ou seja, nesse caso também podemos sortear um elemento o_j de $C_1 \cup C_2$ e examinar se ele pertence a interseção $C_1 \cap C_2$. Se isto for verificado, teremos $max(C_1, h) = max(C_2, h) = o_j$. Caso contrário, o_j será a assinatura máxima somente de um dos conjuntos: $o_j = max(C_1, h) \neq max(C_2, h)$ ou $max(C_1, h) \neq max(C_2, h) = o_j$. A Equação (4.5) que define a similaridade $sim_{jac}(C_1, C_2)$ será, portanto, análoga a (3.5): o numerador $|C_1 \cap C_2|$ da razão representa a quantidade de casos com equivalência de *max-Hashes* e o denominador $|C_1 \cup C_2|$ representa a quantidade dos diferentes possíveis sorteios de o_j.

$$\mathbf{Pr}(max(C_1, h) = max(C_2, h)) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|} = sim_{jac}(C_1, C_2) \tag{4.5}$$

Assim como no caso do *min-Hash*, para o método *max-Hash* também utilizamos N permutações dadas por funções *hash* independentes ($\{h_1, h_2, \cdots, h_N\}$) com intuito de estimar a similaridade $sim_{jac}(C_1, C_2)$. Após o cálculo das assinaturas *max-Hash* segundo cada função h_n, serão gerados "esboços" $\overline{C}_{max,i}$ definidos por (4.6).

$$\overline{C}_{max,i} = (max(C_i, h_1), max(C_i, h_2), \cdots, max(C_i, h_N)) \tag{4.6}$$

A similaridade par a par entre as assinaturas dos conjuntos C_1 e C_2 é calculada através da Equação (4.7).

$$sim_{jac}(\overline{C}_{max,1}, \overline{C}_{max,2}) = \frac{1}{N} \sum_{n=1}^{N} \gamma(\overline{C}_{max,1}(n), \overline{C}_{max,2}(n)), \tag{4.7}$$

Onde $\overline{C}_{max,i}(n)$ é a assinatura *max-Hash* do conjunto C_i segundo a permutação induzida por h_n e

$$\gamma(A, B) = \begin{cases} 1 & se\ A = B \\ 0 & caso\ contrário \end{cases} \tag{4.8}$$

Utilizamos o algoritmo empregado no método *Minwise Hashing* (seção 3.5.4) como base para o desenvolvimento do algoritmo do *Maxwise Hashing*. A computação das assinaturas *max-Hash* também requer a criação da matriz característica M e de *slots* para todo par (h_n, C_i). Novamente, os *slots* têm a finalidade de armazenar a assinatura *max-Hash* de C_i segundo a permutação dada por h_n.

As únicas diferenças no Algoritmo 3 para o cálculo das assinaturas *max-Hash* do conjunto $C_i \in M$ são: 1) A inicialização do algoritmo:

$$\text{Inicialize todo } slot_{max}(h_n, C_i) \text{ com } -\infty$$

2) A modificação do condicional destacado ("Seleção da assinatura"). Para o algoritmo *Maxwise Hashing*, deve-se substituir o conteúdo do *frame* em vermelho no Algoritmo 3 pelas seguintes instruções:

$$\textbf{se } h_n(j) > slot_{max}(h_n, C_i) \textbf{ então}$$
$$slot_{max}(h_n, C_i) \leftarrow h_n(j)$$

4.3.2 *MinMinwise Hashing e MaxMaxwise Hashing*

Baseando-nos na premissa demonstrada por PAGH *et al.* (2014), de que um outro modo de sumarizar conjuntos seja dado por k valores gerados por uma única permutação ao invés de utilizar os menores valores obtidos através de k permutações dos conjuntos, desenvolvemos as abordagens *MinMinwise Hashing* e *MaxMaxwise Hashing*. Estas buscam explorar duas diferentes características dos conjuntos. Genericamente, podemos selecionar dois conjuntos C_1 e C_2 e assim, utilizando o *MinMinwise Hashing*, é possível comparar C_1 e C_2 através da ocorrência de seus dois menores elementos. O método *MaxMaxwise Hashing*, por sua vez, tem por objetivo a comparação dos dois maiores elementos de C_1 e C_2.

Nestas abordagens, tratamos cada um dos limites de cada conjunto individualmente, apenas agregando mais informação a esse limite. Como são produzidas duas dimensões para cada permutação utilizada, esperamos que essas abordagens apresentem resultados melhores dos que os produzidos pelo *Minwise Hashing* e pelo *Maxwise Hashing* para um dado número de permutações. Ou ainda que, para o mesmo resultado, seja necessário permutar os conjuntos um menor número de vezes.

A construção do algoritmo do *MinMinwise Hashing* e do *MaxMaxwise Hashing* é análoga ao *Minwise Hashing* e ao *Maxwise Hashing*, respectivamente. O *MinMinwise Hashing* produz duas assinaturas mínimas referentes a uma permutação aplicada a cada conjunto C_i. Assim, onde no algoritmo *Minwise Hashing* tínhamos

o apenas um *slot* ($slot_{min}$), teremos agora $slot_{1\circ min}$ e $slot_{2\circ min}$ correspondendo ao primeiro e segundo mínimos, respectivamente.

De modo semelhante, para o *MaxMaxwise Hashing*, as permutações geram duas assinaturas máximas para cada conjunto C_i. Portanto, teremos também dois *slots* ($slot_{1\circ max}$ e $slot_{2\circ max}$) criados com o objetivo de representar o maior e o segundo maior valor produzido para elementos de C_i por h.

4.3.3 *MinMaxwise Hashing*

Outra abordagem que decidimos adotar no presente trabalho foi utilizar tanto o limite inferior quanto o limite superior para delimitar o conjunto que representa cada documento analisado. Desta forma, uma vez que se observa tanto o máximo quanto o mínimo de um conjunto, espera-se que os resultados obtidos com essa abordagem, que chamaremos de *MinMaxwise Hashing*, sejam melhores que aqueles produzidos pelo *Minwise* e pelo *Maxwise Hashing*. Neste método, buscaremos explorar o conceito de reticulado, estabelecendo simultaneamente as fronteiras inferiores e superiores dos conjuntos analisados.

A grande vantagem do *MinMaxwise Hashing* em relação às já citadas abordagens é que podemos captar mais informações para representar os conjuntos utilizando a mesma quantidade de permutações. Portanto, verificamos que esse método é mais rígido que os anteriores visto que, para serem considerados semelhantes, dois conjuntos tenham que ter limites coincidentes. Por esse motivo, é também esperado que a quantidade de conjuntos erroneamente classificados como semelhantes, os falsos positivos, diminua.

O *MinMaxwise Hashing* é capaz de agregar as propriedades (3.4) e (4.4) definidas para o *Minwise* e o *Maxwise Hashing*, respectivamente. Geramos, para esse método, duas assinaturas *minmax-Hash* referentes à permutação dada por uma função *hash* h, sendo uma delas a assinatura mínima do conjunto, correspondente ao *min-Hash*, e outra assinatura, correspondente ao máximo do conjunto, o *max-Hash*.

As assinaturas *minmax-Hash* são obtidas através de N funções *hash* independentes ($\{h_1, h_2, \cdots, h_N\}$), assim como nas abordagens *Minwise Hashing* e *Maxwise Hashing*. Os esboços, porém, contêm o dobro da informação em relação àquela extraída nessas abordagens. Cada conjunto C_i é caracterizado por um "esboço" $\overline{C}_{minmax,i}$. Os "esboços" gerados para o *MinMaxwise Hashing* são dados por (4.9).

$$\overline{C}_{minmax,i} = \overline{C}_{min,i} \parallel \overline{C}_{max,i} \tag{4.9}$$

No *minmax-Hash*, os "esboços" utilizados nada mais são do que uma concatenação dos vetores correspondentes aos "esboços" representados por (3.6) e (4.6), calculados para o *min-Hash* e para o *max-Hash*, respectivamente.

Após a obtenção dos "esboços", podemos estimar a similaridade par a par entre as assinaturas de dois conjuntos genéricos C_1 e C_2 através da Equação (4.7).

$$sim_{jac}(\overline{C}_{minmax,1}, \overline{C}_{minmax,2}) = \frac{1}{N}\sum_{n=1}^{N}\gamma(\overline{C}_{minmax,1}(n), \overline{C}_{minmax,2}(n)) \tag{4.10}$$

Onde $\overline{C}_{minmax,i}(n)$ é a assinatura *minmax-Hash* gerada para o conjunto C_i por meio da permutação dada por h_n e

$$\gamma(A, B) = \begin{cases} 1 & se\ A = B \\ 0 & caso\ contrário \end{cases} \tag{4.11}$$

O algoritmo do método *MinMaxwise Hashing* consiste em uma adaptação dos outros métodos já apresentados, mas conservando algumas propriedades comuns também às abordagens *Minwise Hashing* e *Maxwise Hashing*, como a criação da matriz característica M, conforme explicado na seção 3.5.3.

Uma das diferenças do algoritmo *minmax-Hash* é a adição de novos *slots*, que permitirão armazenar as assinaturas mínima e máxima para cada conjunto C_i de acordo com a permutação dada por uma função *hash* h_n. Ou seja, para todo par (h_n, C_i), devem ser criados dois *slots*, um para o mínimo ($slot_{min}$) e outro para o máximo ($slot_{max}$).

Tendo por base o Algoritmo 3, pequenas modificações devem ser feitas. A primeira delas é a inicialização, que deve criar os *slots* usados no algoritmo do *MinMaxwise Hashing*:

$$\text{Inicialize todo } slot_{min}(h_n, C_i) \text{ com } \infty$$
$$\text{Inicialize todo } slot_{max}(h_n, C_i) \text{ com } -\infty$$

Outra alteração realizada no Algoritmo 3 para o cálculo do *minmax-Hash* do conjunto $C_i \in M$ é a substituição do condicional destacado em vermelho ("Seleção da assinatura") pelo seguinte trecho de código:

$$\textbf{se } h_n(j) < slot_{min}(h_n, C_i) \textbf{ então}$$
$$slot_{min}(h_n, C_i) \leftarrow h_n(j)$$

$$\textbf{se } h_n(j) > slot_{max}(h_n, C_i) \textbf{ então}$$
$$slot_{max}(h_n, C_i) \leftarrow h_n(j)$$

4.4 Formalização dos métodos propostos

Com o intuito de analisar os métodos propostos na seção 4.3, utilizamos o *corpus* METER, apresentado na seção 2.2. O estudo realizado neste trabalho busca descobrir quais notícias produzidas pela Press Association dão origem a cada documento categorizado como Completamente Derivado ou Parcialmente Derivado dentro do *corpus*. Na análise desses casos de reúso, foram utilizados tanto os documentos pertencentes ao domínio corte Britânica quanto aqueles que se referem a notícias do *show business*.

Para avaliar as abordagens *Minwise Hashing*, *Maxwise Hashing*, *MinMaxwise Hashing*, *MinMinwise Hashing* e *MaxMaxwise Hashing*, inicialmente dividimos o *corpus* METER em dois grupos. O primeiro grupo, chamado "Produzidos" se refere às matérias que foram criadas pela Press Association. Esse grupo contém documentos que são considerados fontes de notícias. O segundo grupo, chamado "Publicados" engloba todos os artigos publicados em jornais, sendo ou não derivados de uma das notícias pertencentes ao primeiro grupo. Assim sendo, nossa meta é identificar a relação entre elementos dos grupos "Produzidos" e "Publicados" e, para isso, definimos um *framework* composto das etapas apresentadas seção 4.4.1

4.4.1 *Framework* de aplicação de famílias de técnicas LSH

O *framework* proposto para analisar o relacionamento entre diferentes documentos é introduzido através do fluxograma representado pela Figura 4.1. A sequência de passos na tarefa de comparação do conteúdo de documentos propostas neste trabalho é (*1*) Tokenização; (*2*) Geração de *fingerprints*; (*3*) Permutação da matriz característica; (*4*) Aplicação da função de seleção; (*5*) Cálculo da similaridade.

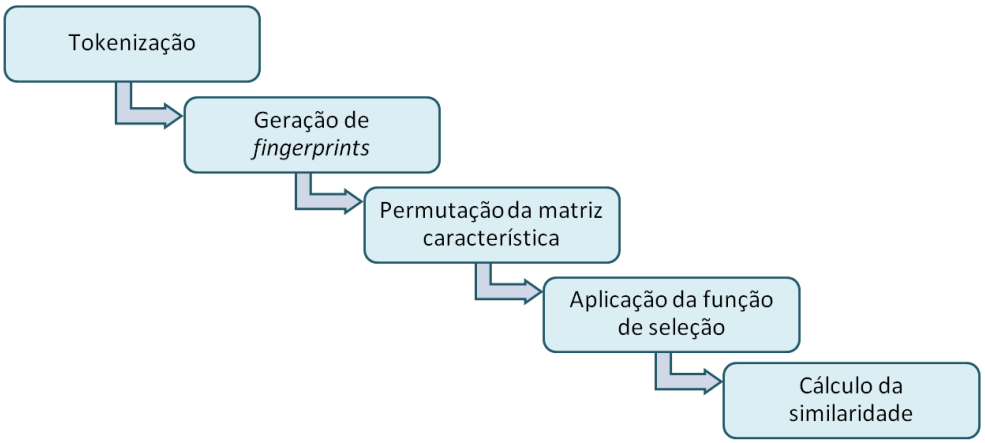

Figura 4.1: Fluxograma proposto

Diversas abordagens podem ser comparadas para cada passo do *framework*. Por

exemplo, diferentes funções de permutação, funções de seleção ou *operadores* podem ser analisados.

1ª etapa: Tokenização. Essa etapa compreende a extração dos termos encontrados em cada documento de ambos os grupos e a construção da sua matriz característica, também chamada de matriz termo-documento, que registra a incidência dos termos nos documentos. Formalmente, definimos:

- D como o conjunto de todos os documentos do *corpus*.
- $d_i \in D$ como o i-ésimo documento de D.
- T como o conjunto de todos os termos presentes no *corpus*.
- $t_j \in T$ como o j-ésimo termo de T.

A tokenização é dividida em dois passos:

(a) Transformação dos documentos em conjuntos (Equação (4.12)).

$$D \rightarrow C, \tag{4.12}$$

onde C_i é o conjunto de termos de d_i

(b) Criação da matriz característica M, onde cada coluna de M é dada por um conjunto $C_i \subset C$, cada linha é dada por um termo $t_j \in T$ e as células $M_{i,j}$ são representações booleanas da incidência dos termos nos conjuntos C_i.

2ª etapa: Geração de *fingerprints*. A segunda etapa corresponde à geração de uma *fingerprint* g para cada termo encontrado no *corpus*. Logo, considerando g_j a *fingerprint* referente ao termo t_j, ela é calculada através da transformação $\psi(t_j)$, definida pela Equação (4.13).

$$\psi(t_j) : T \rightarrow \mathbb{R} \tag{4.13}$$

No caso deste trabalho, utilizou-se o algoritmo MD5 (RIVEST, 1992) como transformação $\psi(t_j)$, mas a transformação poderia ter sido realizada através de outro método, como o algoritmo de implementação de *fingerprints* proposto por KARP e RABIN (1987) e o *Secure Hash Algorithm* (SHA) (EASTLAKE e JONES, 2001).

3ª etapa: Permutação da matriz característica. A função de permutação consiste em um método para realizar as permutações das linhas da matriz característica exigidas pelos métodos propostos. Assim, sejam:

- G, como o conjunto de todas as *fingerprints* g_j produzidas por $\psi(t_j)$ para os termos $t_j \in T$ e;
- L, a sequência formada pelas *fingerprints* $g_j \in G$ na ordem determinada pelas linhas de M.

A função de permutação $\pi_n(L)$ é definida pela Equação (4.14).

$$\pi_n(L) : L \to L \tag{4.14}$$

Cada função de permutação $\pi_n(L)$ leva à construção de uma nova matriz característica M_n, que será utilizada na próxima tarefa prevista no fluxograma. As permutações das linhas de M são realizadas por meio de funções *hash* projetadas para reordenar L de forma aleatória no início da execução.

Considerando, por exemplo, de modo abstrato, dois documentos d_1 e d_2, representados, respectivamente, pelos conjuntos $C_1 = \{a, b\}$ e $C_2 = \{a, b, d\}$ em um universo $\Omega = \{a, b, c, d, e\}$, escolhemos ao acaso duas permutações π_1 e π_2. A ordem induzida pela função de permutação π_1 é *abcde*, enquanto que a ordem induzida pela função π_2 é *edcba*, como pode ser observado na tabela 4.2

	$n = 1$	$n = 2$
$\pi_n(a)$	1	5
$\pi_n(b)$	2	4
$\pi_n(c)$	3	3
$\pi_n(d)$	4	2
$\pi_n(e)$	5	1

Tabela 4.2: Ordens induzidas pelas permutações π_1 e π_2.

A representação gráfica das permutações π_1 e π_2 aplicadas aos conjuntos C_1 e C_2 é mostrada na Figura 4.2.

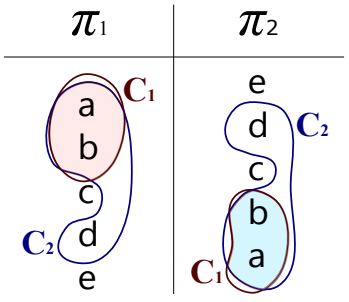

Figura 4.2: Representação gráfica das permutações π_1 e π_2 aplicadas aos conjuntos C_1 e C_2.

Optamos por empregar como função de permutação, o algoritmo randômico *Universal Hashing* (CARTER e WEGMAN, 1977). Este algoritmo seleciona ao acaso

uma função *hash* a partir de uma "classe de funções cuidadosamente projetada" (CORMEN *et al.*, 2009). A aleatoriedade do *Universal Hashing* permite um comportamento diferente em execuções distintas, ainda que seja utilizada a mesma entrada (CORMEN *et al.*, 2009). Por isso, ele assegura uma baixa probabilidade de colisão entre os valores gerados para entradas distintas quando emprega-se a mesma função *hash* (STINSON, 1994).

4ª etapa: Aplicação da função de seleção. É na quarta etapa que aplicamos o método que irá escolher os elementos representativos de cada conjunto $C_i \subset C$. A função de seleção varia de acordo com a abordagem que adotamos e, consequentemente, com os *operadores* que pretendemos explorar.

Ainda considerando o exemplo apresentado na etapa anterior, onde se utiliza as funções de permutação π_1 e π_2, a seleção dos elementos representativos de C_1 e C_2 pode ser feita através da escolha do mínimo, do máximo ou de um outro *operador* qualquer. A figura 4.3 ilustra a seleção dos *operadores* mínimo e máximo de acordo com a permutação π_1 (Figura 4.3 (a)) e segundo a permutação π_2 (Figura 4.3 (2)).

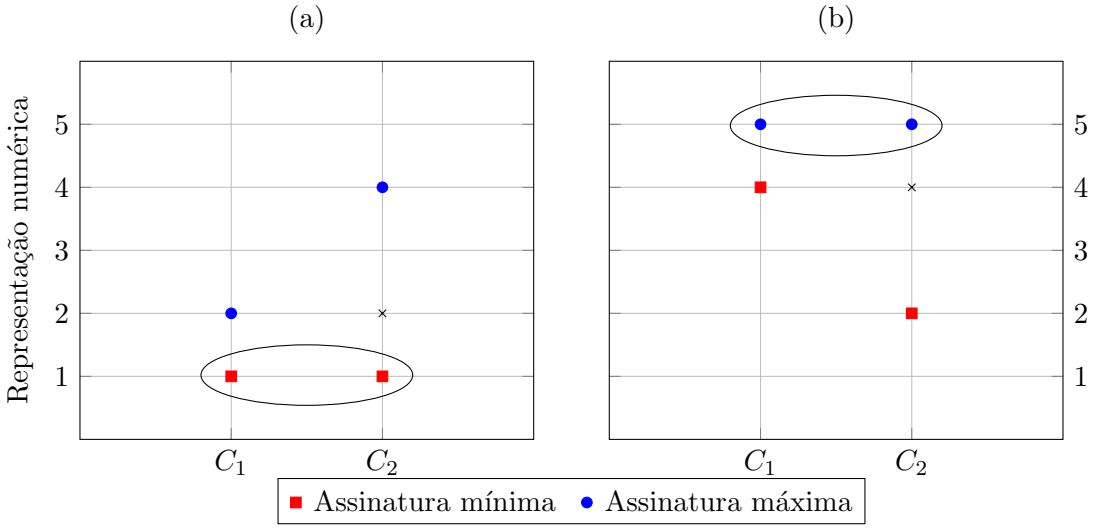

Figura 4.3: Assinaturas dos conjuntos C_1 e C_2 de acordo com (a) a função de permutação π_1 e (b) a função de permutação π_2.

Como pode ser observado, os conjuntos C_1 e C_2 apresentam o mesmo mínimo quando escolhemos a permutação π_1 e apresentam o mesmo máximo quando realizamos a permutação π_2.

Portanto, para cada conjunto $k = min$, será utilizado o *operador* mínimo (min_1) por meio da função de seleção μ para a escolha da menor *fingerprint* g_j; $k = max$, será utilizado o *operador* máximo (max_1) por meio da função de seleção μ para a escolha da maior *fingerprint* g_j; $k = minMax$, serão utilizados os *operadores* mínimo (min_1) e máximo (max_1) por meio da função de seleção μ para a escolha da menor e da maior *fingerprint* de cada conjunto; $k = minMin$, serão utilizados os *operadores*

correspondentes aos dois menores elementos (min_1 e min_2) por meio da função de seleção μ para a escolha das duas menores *fingerprints* de cada conjunto; e $k = maxMax$, serão utilizados os *operadores* correspondentes aos dois maiores elementos (max_1 e max_2) por meio da função de seleção μ para a escolha das duas maiores *fingerprints* de cada conjunto; como pode ser observado na Tabela 4.1.

Definimos a assinatura $s_{i,n,k}$ (Equação (4.15).)de um conjunto C_i (uma coluna C_i) calculada com base na permutação π_n de seus elementos, que são escolhidos através de uma função de seleção genérica μ baseada no conjunto de *operadores* de seleção $O_k \subset O$.

$$s_{i,n,k} = \mu(C_i, \pi_n(L), O_k) \tag{4.15}$$

As assinaturas $s_{i,n,k}$ correspondem aos vetores formados pelos valores numéricos referentes às funções de seleção aplicadas a cada permutação π_n do conjunto C_i de acordo com o conjunto de *operadores* O_k. A Equação (4.16) mostra o velor de seleção de assinaturas $s_{i,n,k}$ para a permutação (n) de um conjunto (i) baseada no conjunto de *operadores* selecionados (k). Cada dimensão trata uma *fingerprint* selecionada correspondente a cada *operador*.

$$s_{i,n,k} = (s^1_{i,n,k}, s^2_{i,n,k}, \cdots, s^P_{i,n,k}). \tag{4.16}$$

Após a execução do método escolhido, é produzida a matriz de assinaturas S do *corpus*, que reúne as assinaturas $s_{i,n,k}$ de todas as permutações π_n dos conjuntos C_i, de acordo com o conjunto de *operadores* (O_k) selecionados. Assim, considerando que existam I documentos e, consequentemente, I conjuntos, dadas N permutações, para P *operadores* selecionados, podemos construir a matriz de assinaturas S (como, por exemplo, na tabela 4.3) com base em um k genérico pré-estabelecido. Cada coluna da matriz S será a representação de um documento no espaço de representação de assinaturas.

5ª etapa: Cálculo da similaridade. Nesta etapa, calculamos a similaridade $sim_m(C_1, C_2)$ (Equação 4.17) entre cada par de conjuntos (C_1, C_2) constituído de um elemento do grupo "Produzidos" e de outro elemento do grupo "Publicados", de acordo com a métrica m.

$$sim_m(C_1, C_2) = S \times S \rightarrow \mathbb{R} \tag{4.17}$$

Embora a métrica padrão para o cálculo da similaridade nas abordagens utilizadas seja o índice de Jaccard, também utilizamos para os testes realizados, a similaridade do cosseno e o coeficiente de Dice. O índice de Jaccard e a similaridade do cosseno são explicados nas seções 3.5 e 3.1, respectivamente.

		i = 1	i = 2	⋯	i = I
n = 1	p = 1	$s^1_{1,1,k}$	$s^1_{2,1,k}$		$s^1_{I,1,k}$
	p = 2	$s^2_{1,1,k}$	$s^2_{2,1,k}$		$s^2_{I,1,k}$
	⋮			⋱	
	p = P	$s^P_{1,1,k}$	$s^P_{2,1,k}$		$s^P_{I,1,k}$
n = 2	p = 1	$s^1_{1,2,k}$	$s^1_{2,2,k}$		$s^1_{I,2,k}$
	p = 2	$s^2_{1,2,k}$	$s^2_{2,2,k}$		$s^2_{I,2,k}$
	⋮			⋱	
	p = P	$s^P_{1,2,k}$	$s^P_{2,2,k}$		$s^P_{I,2,k}$
⋮			⋱		
n = N	p = 1	$s^1_{1,N,k}$	$s^1_{2,N,k}$		$s^1_{I,N,k}$
	p = 2	$s^2_{1,N,k}$	$s^2_{2,N,k}$		$s^2_{I,N,k}$
	⋮			⋱	
	p = P	$s^P_{1,N,k}$	$s^P_{2,N,k}$		$s^P_{I,N,k}$

Tabela 4.3: Construção da matriz de assinaturas S para N permutações de I conjuntos através de um conjunto de *operadores* O_k, com P *operadores*.

O coeficiente de Dice (DICE, 1945) medido para dois conjuntos C_1 e C_2 é calculado através da razão entre o dobro da interseção $C_1 \cap C_2$ e soma do tamanho dos dois conjuntos (Equação 4.18) (MANNING *et al.*, 2008).

$$sim_{dic}(C_1, C_2) = \frac{2\,|C_1 \cap C_2|}{|C_1| + |C_2|}, \tag{4.18}$$

Onde o fator *2* permite obter um valor variando entre *0* e *1* (BARRÓN-CEDEÑO, 2010, MANNING *et al.*, 2008).

4.4.2 Análise de complexidade dos métodos propostos

Uma vez formalizados os métodos propostos, é possível calcular a complexidade computacional de cada um deles e realizar a sua comparação. Nesta seção, estudamos a complexidade da operação de extração das assinaturas de um único conjunto de elementos C_i contido no universo de todos os termos (T). Assim, considerando T com cardinalidade igual a $|T|$, (ou seja, a $|T|$ é quantidade de todos os diferentes termos do *corpus*) e a quantidade de permutações utilizadas no problema igual a N, o cálculo da complexidade do algoritmo *Minwise Hashing* é dado por $O(|T| \times N)$.

A complexidade do algoritmo *Maxwise Hashing* pode ser calculada de modo

análogo, e também tem $O(|T| \times N)$. Ambos os algoritmos (*Minwise* e *Maxwise Hashing*) apresentam a mesma complexidade computacional pois a apenas a função de seleção é alterada, sem a criação de novos laços (*loops*).

No caso dos algoritmos *MinMinwise Hashing*, *MinMaxwise Hashing* e *MaxMaxwise Hashing*, como são utilizadas duas funções de seleção distintas para cada um deles, suas respectivas complexidades computacionais são dadas por $O(|T| \times N \times 2)$. Neste caso, dois *loops* são implementados para tornar possível o cálculo da assinatura de C_i.

Genericamente, um algoritmo que estuda P diferentes *operadores* tem a complexidade igual a $O(|T| \times N \times P)$. Esse algoritmo representa, portanto, um conjunto C_i por meio de p assinaturas para cada função de permutação π_n. O uso de mais *operadores* para representar os conjuntos acarreta também o aumento da informação por ele agregada às suas assinaturas. Logo, é esperado que, à medida que se tenha mais assinaturas para um conjunto, os métodos utilizados atinjam melhores resultados ao identificar documentos semelhantes.

Capítulo 5

Resultados e Discussões

5.1 Metodologia para avaliação dos métodos propostos

As consultas realizadas no presente trabalho têm por objetivo analisar os resultados gerados pelos métodos *Minwise Hashing*, *Maxwise Hashing*, *MinMaxwise Hashing*, *MinMinwise Hashing* e *MaxMaxwise Hashing* na tarefa de identificação de reúso de texto entre notícias produzidas pela Press Association (grupo "Produzidos") e artigos publicados em jornais britânicos (grupo "Publicados").

Conhecemos, a priori, todas as relações existentes entre elementos de ambos os grupos. E assim, definimos como relevantes para uma consulta c, feita para um elemento j do grupo "Produzidos", os itens d_i do grupo "Publicados" que foram mapeados no *corpus* METER como completamente ou parcialmente derivados de i. Ou seja, uma consulta equivale à identificação das matérias fontes produzidas pela Press Association para cada texto publicado pelo conjunto de jornais analisados no projeto METER.

Ao executar a abordagem escolhida, esperamos detectar casos de reúso através da análise da similaridade entre objetos dos diferentes grupos de documentos. Portanto, acreditamos que os artigos relevantes para a consulta sejam similares às notícias da PA a partir das quais foram produzidos.

Vimos anteriormente que o produto final do nosso *framework* é a similaridade par a par de documentos dos grupos "Produzidos" e "Publicados". Portanto, devemos agora avaliar cada abordagem utilizada, para que possamos compará-las e observar qual é mais vantajosa para o tipo de problema estudado.

5.2 Métricas de Avaliação

Segundo BARRÓN-CEDEÑO (2010), em tarefas de Recuperação de Informação, existem dois objetivos principais. O primeiro consiste em recuperar a maior quantidade possível de documentos considerados relevantes e, o segundo objetivo se refere a recuperar a menor quantidade possível de documentos considerados não relevantes.

Duas medidas são conhecidas por representar esses objetivos: a precisão e a revocação. A precisão (Equação (5.1)) é definida como a razão entre o número de documentos que são relevantes e o total de documentos recuperados em uma busca (MANNING *et al.*, 2008).

$$Precisão = \frac{\#itens\ relevantes\ recuperados}{\#itens\ recuperados} \tag{5.1}$$

A revocação (Equação (5.2)), por outro lado, é definida como a razão entre o número de documentos relevantes recuperados em uma busca e o total de documentos relevantes (MANNING *et al.*, 2008).

$$Revocação = \frac{\#itens\ relevantes\ recuperados}{\#itens\ relevantes} \tag{5.2}$$

5.3 Precisão Interpolada

Os sistemas de Recuperação de Informação pretendem sempre melhorar as taxas de precisão e revocação no conjunto de documentos recuperados (DE ANDRADE LEITE, 2009). Para isso, é comum que os documentos sejam ordenados e apresentados de acordo com um critério de relevância (DE ANDRADE LEITE, 2009). Cada conjunto de documentos recuperados pode ser plotado em um gráfico através dos seus valores de precisão e revocação e, a esse gráfico, chamamos de curva precisão e revocação (MANNING *et al.*, 2008).

Para facilitar e padronizar a análise dos algoritmos cujas consultas recuperam conjuntos com diferentes quantidades de elementos relevantes, os valores de precisão calculados para cada documento são interpolados (BAEZA-YATES *et al.*, 1999). A precisão interpolada em dado nível de revocação r é definida como a maior precisão encontrada para qualquer nível de revocação $r' \geq r$ (5.3) (MANNING *et al.*, 2008).

$$Precisão\ interpolada = max_{(r' \geq r)} Precisão(r') \tag{5.3}$$

MANNING *et al.* (2008) explicam que a curva precisão e revocação é extremamente informativa, mas, comumente, em problemas de Recuperação de Informação, deseja-se resumi-la. Os autores também apresentam em (MANNING *et al.*, 2008) a maneira tradicional de fazer isto por meio da representação da precisão por 11 níveis

de revocação ("11 *standard recall levels*"), em que se mede a precisão interpolada para os seguintes níveis de revocação $(0\%, 10\%, 20\%, \cdots, 100\%)$. Posteriormente, calcula-se a média aritmética da precisão interpolada em cada nível de revocação para cada consulta realizada.

Como exemplo, consideramos duas consultas c_1 e c_2 cujos resultados relevantes são dados pelos conjuntos $C_{1,rel} = \{a_3, a_7, a_9, a_{11}, a_{15}\}$ e $C_{2,rel} = \{a_2, a_3, a_9, a_{10}\}$. Suponha que os conjuntos de documentos recuperados para as consultas c_1 e c_2 são, respectivamente, $C_{1,rec} = \{a_7, a_1, a_{11}, a_8, a_6, a_9, a_{12}, a_4, a_2, a_3\}$ e $C_{2,rec} = \{a_8, a_{10}, a_7, a_3, a_6\}$. A precisão interpolada das consultas é dada pela Tabela 5.1. Também podemos representar a precisão em 11 níveis de revocação através do Gráfico 5.1.

	Precisão Interpolada	
Revocação	c_1	c_2
0%	100%	100%
10%	100%	50%
20%	100%	50%
30%	67%	50%
40%	67%	50%
50%	50%	50%
60%	50%	50%
70%	40%	50%
80%	40%	50%
90%	40%	50%
100%	40%	50%

Tabela 5.1: Precisão interpolada das consultas c_1 e c_2.

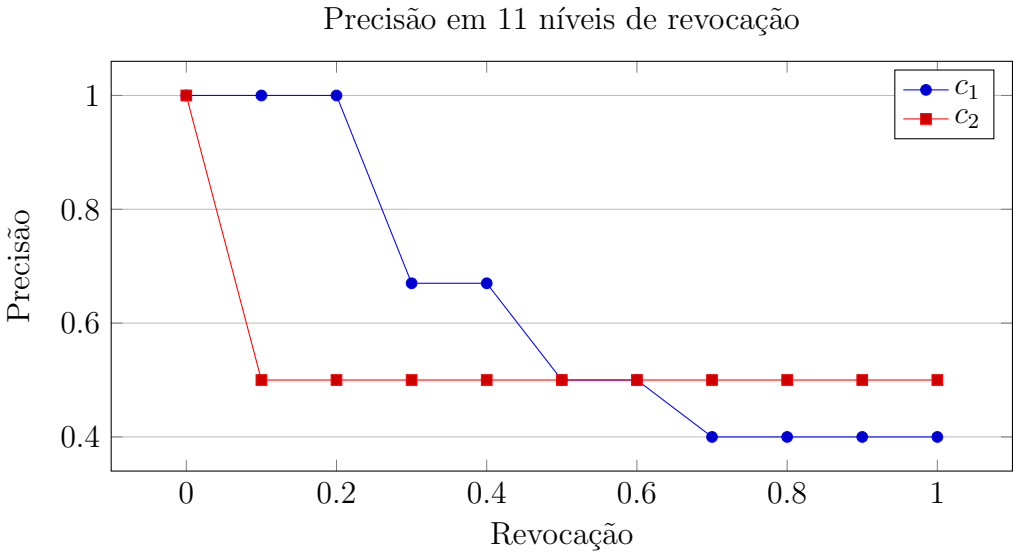

Figura 5.1: Precisão em 11 níveis de revocação.

Através dos valores obtidos para a precisão interpolada de c_1 e c_2, também podemos gerar um gráfico da precisão média das duas consultas (Gráfico 5.2). As curvas médias de precisão e revocação são usadas para comparar os resultados de diferentes algoritmos de recuperação (BAEZA-YATES *et al.*, 1999).

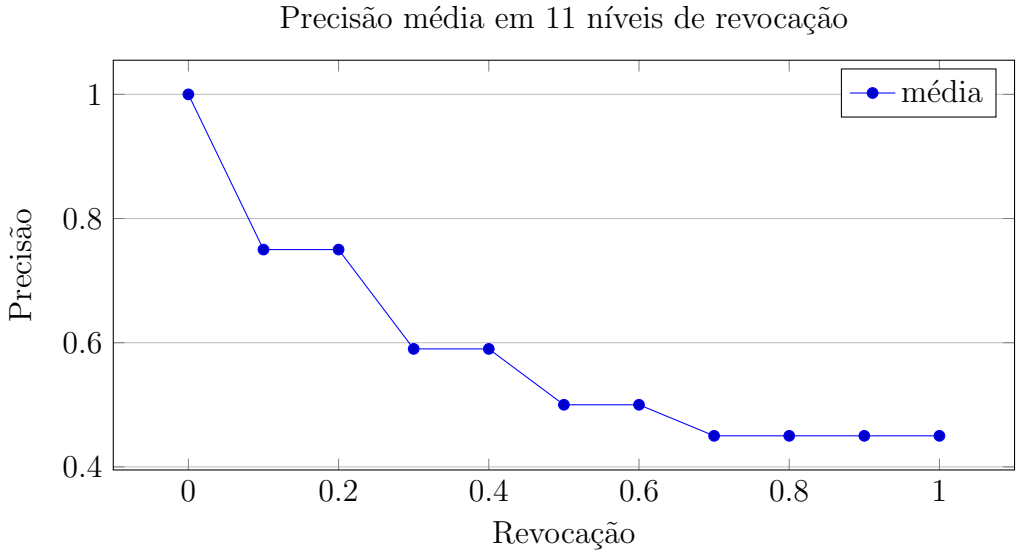

Figura 5.2: Precisão média em 11 níveis de revocação.

Ao avaliar os nossos experimentos, optamos por utilizar como métrica a precisão média interpolada em 11 níveis de revocação. Entretanto, como realizamos testes para diferentes quantidades de permutações nos algoritmos *Minwise Hashing*, *Maxwise Hashing*, *MinMaxwise Hashing*, *MinMinwise Hashing* e *MaxMaxwise Hashing*, optamos também por, para cada análise de uma combinação de técnica e número de permutações, calcular a área sob a curva da precisão média em 11 níveis de revocação. Assim, cada execução passa a ser representada por um único valor (a área sob a curva) e podemos visualizar em um único gráfico os resultados das diferentes combinações adotadas.

5.4 Experimentos

Neste trabalho foram realizadas execuções para as abordagens *Minwise*, *Maxwise*, *MinMaxwise*, *MinMinwise* e *MaxMaxwise Hashing*, variando-se a quantidade de permutações utilizadas em valores de potências 2^n para $n \in \{1, 2, \cdots, 10\}$. Também foram utilizadas como métricas de similaridade, o índice de Jaccard, a similaridade do cosseno e o coeficiente de Dice.

Em cada execução, procuramos identificar os artigos produzidos pela PA que deram origem às matérias publicadas pertenteces ao grupo "Publicados". Ou seja, uma execução engloba o conjunto de consultas realizadas no corpus METER, consistindo em um total de 944 consultas.

Como *baseline*, utilizamos a técnica *Bag of Words (BoW)* que, segundo BARRÓN-CEDEÑO (2010), é uma das representações mais frequentemente usadas em modelos de Recuperação da Informação. Nesta abordagem, a ocorrência de cada termo é contada, a despeito de sua ordem (MANNING *et al.*, 2008).

É importante ressaltar que as técnicas empregadas não pretendem obter melhores resultados do que o *Bag of Words* pois, como toda técnica de *Locality-Sensitive Hashing*, tratam o problema de redução da dimensionalidade, ficando assim sujeitas a perdas de qualidade.

Nas execuções realizadas utilizando o *Bag of Words*, optamos por selecionar os 2^n *tokens* mais relevantes para o documento, considerando $n \in \{1, 2, \cdots, 10\}$. A seleção dos *tokens* mais relevantes é feita através da construção de um vocabulário através da tokenização do documento e da posterior seleção dos seus 2^n *tokens* de maior frequência. Desta forma, os resultados de cada execução do BoW são calculados utilizando o mesmo número de dimensões (a mesma quantidade de informação) dos métodos propostos.

Os resultados obtidos estão representados nos gráficos 5.3, 5.4, 5.5, agrupados para cada métrica de similaridade. Optamos por representar o eixo horizontal dos gráficos como a quantidade de dimensões utilizadas em cada método (conforme explicado na seção 4.4). Trabalhamos assim para podermos comparar os resultados gerados para as abordagens que utilizam mais de um *operador* para representar os documentos em relação às outras duas também estudadas (*Minwise Hashing* e *Maxwise Hashing*). No caso dos métodos *MinMaxwise Hashing*, *MinMinwise Hashing* e *MaxMaxwise Hashing*, cada permutação explora dois *operadores* dos conjuntos (o mínimo e o máximo, os dois mínimos ou os máximos, respectivamente), produzindo duas dimensões, enquanto para o *Minwise Hashing* e para o *Maxwise Hashing*, as seleções são feitas considerando apenas um *operador* (o mínimo ou o máximo, respectivamente), e consequentemente uma dimensão.

No eixo vertical dos gráficos, representamos a área sob a curva (ASC) da precisão média em 11 níveis de revocação. Portanto, nesse caso, teremos valores variando de 0 a 1.

Como pode ser constatado nos três gráficos, as curvas obtidas para o *Minwise Hashing* e para o *Maxwise Hashing* são muito próximas, apresentando resultados bastante similares para no caso das três métricas de similaridade adotadas e da quantidade de dimensões avaliadas. O mesmo acontece com as curvas das abordagens *MinMinwise Hashing* e *MaxMaxwise Hashing*, levando-nos a concluir que, nesses casos, há uma melhoria linear no comportamento dos resultados em relação ao número de permutações estudadas. Ou seja, para a mesma quantidade de dimensões, visto que exploram dois diferentes *operadores* dos conjuntos, os métodos *MinMinwise Hashing* e *MaxMaxwise Hashing* necessitam de apenas metade do número de per-

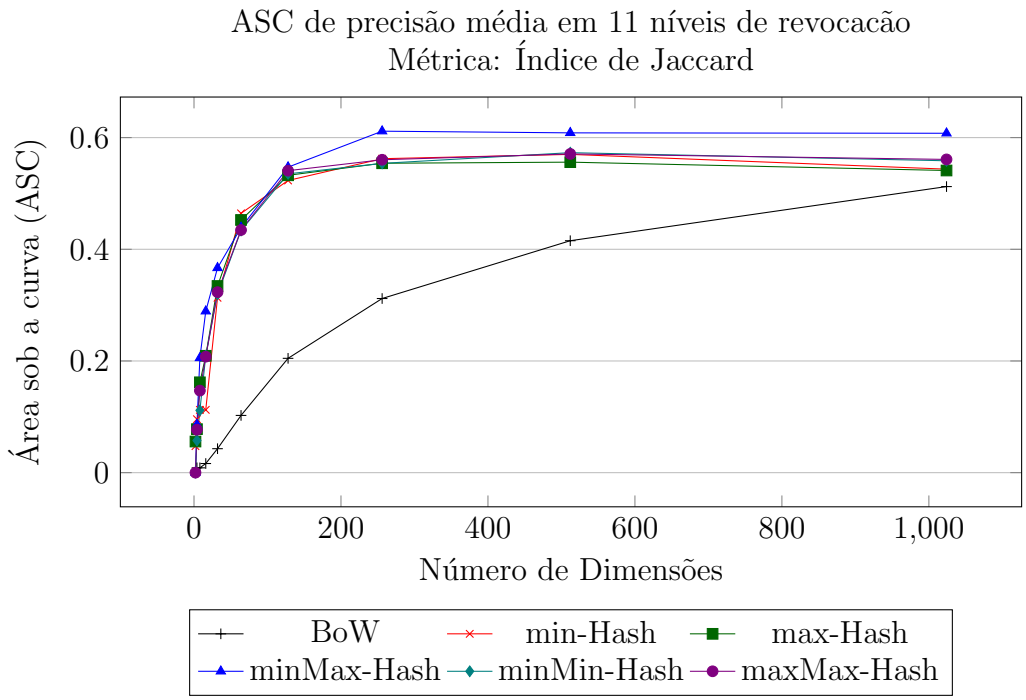

Figura 5.3: Área sob a curva de precisão média em 11 níveis de revocação ao utilizar o índice de Jaccard.

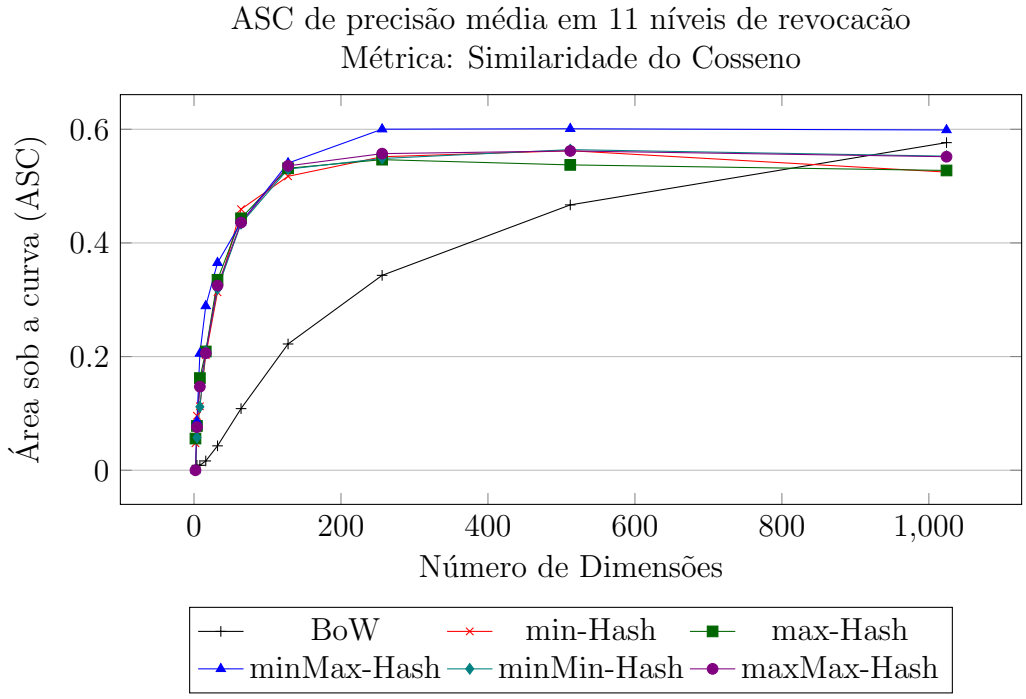

Figura 5.4: Área sob a curva de precisão média em 11 níveis de revocação ao utilizar a similaridade do cosseno.

mutações para produzir os mesmos resultados do *Minwise* e do *Maxwise Hashing*.

Ao comparar os resultados do *MinMaxwise Hashing* com aqueles produzidos para as outras abordagens LSH estudadas, observa-se a superioridade do primeiro em todas as três métricas de similaridade, se destacando em praticamente todos os

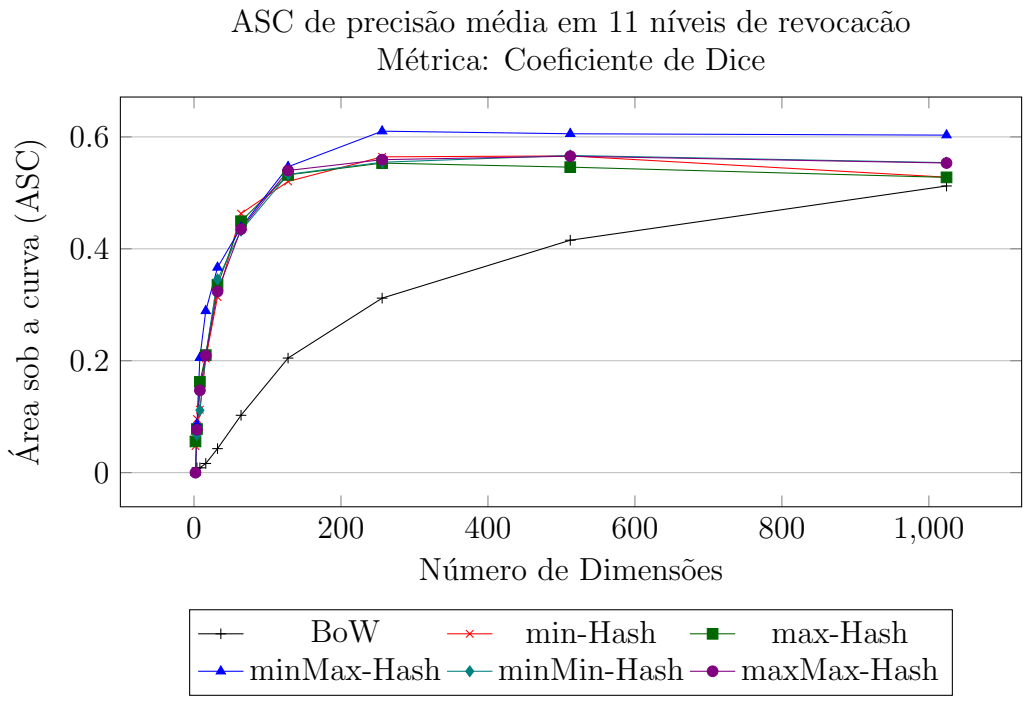

Figura 5.5: Área sob a curva de precisão média em 11 níveis de revocação ao utilizar o coeficiente de Dice

casos analisados para diferentes números de dimensões, mas principalmente para dimensões maiores (a partir de 2^7 dimensões). A isso atribuímos a baixa representatividade do conteúdo de um documento que poucas dimensões produzem. Assim, é possível constatar que, para a mesma quantidade de dimensões, consequentemente, menor quantidade de permutações, o *MinMaxwise Hashing* é capaz de alcançar resultados melhores dos que os gerados para as abordagens *Minwise* e *Maxwise Hashing* na tarefa de identificação de reúso. Isso indica que o *MinMaxwise Hashing* tem uma melhor capacidade de representar conjuntos do que as outras abordagens, apresentando, inclusive, uma melhoria superlinear quando considerada apenas a quantidade de permutações realizadas.

No entanto, para pequenos valores de 2^n, isto é, poucas dimensões consideradas no experimento, as cinco abordagens de *Locality-Sensitive Hashing* apresentam resultados muito superiores ao *Bag of Words*. Ou seja, isso mostra que o *Minwise Hashing*, o *Maxwise Hashing*, o *MinMaxwise Hashing*, o *MinMinwise Hashing* e o *MaxMaxwise Hashing* conseguem representar os documentos de forma muito mais próxima da realidade quando utilizamos poucas dimensões.

Também podemos constatar através dos gráficos gerados que as métricas de similaridade utilizadas têm pouco impacto nos resultados das abordagens de *Locality-Sensitive Hashing*, ainda que o índice de Jaccard apresente um leve ganho em relação às outras métricas utilizadas. O mesmo não se verifica para a abordagem *Bag of Words*, que é claramente beneficiada pela similaridade do cosseno.

Através do conjunto de gráficos de precisão média em 11 níveis de revocação apresentados na Figura 5.6, podemos constatar a superioridade da abordagem *Min-Maxwise Hashing* em todas as três métricas avaliadas (Similaridade do Cosseno, Índice de Jaccard e Coeficiente de Dice). Os gráficos mostram os resultados dos métodos estudados considerando-se experimentos de identificação de reúso de texto no *corpus* METER com 1024 (2^{10}) dimensões.

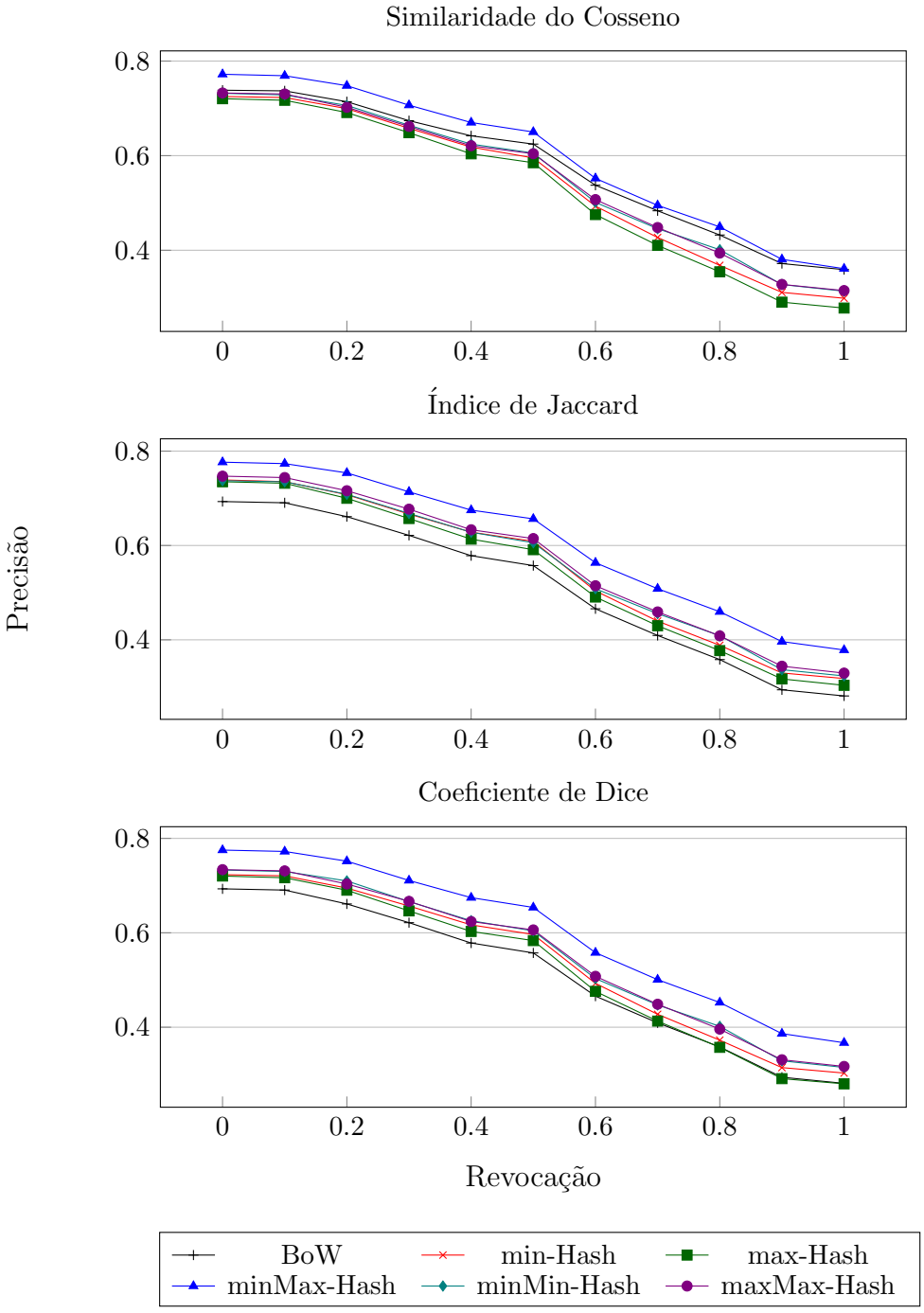

Figura 5.6: Precisão média em 11 níveis de revocação para 1024 dimensões. Experimentos usando as métricas: Similaridade do Cosseno, Índice de Jaccard e Coeficiente de Dice.

Os gráficos da Figura 5.6 também confirmam que os valores de precisão interpolada para as abordagens *Minwise* e *Maxwise Hashing* são muito próximos, assim como os das abordagens *MinMinwise Hashing* e o *MaxMaxwise Hashing*. Porém, essas últimas, assim como o *MinMaxwise Hashing*, necessitam de metade das operações de permutação para alcançar os mesmos resultados.

Com o uso de 1024 dimensões, o *baseline Bag of Words* obtém resultados mais próximos aos das abordagens estudadas pricipalmente com a similaridade do cosseno; tendo, por outro lado, os piores resultados quando empregamos como métrica o índice de Jaccard. Isto nos diz que, enquanto o BoW é mais recomendado para maiores dimensões com o uso da medida do cosseno para calcular similaridade, o índice de Jaccard leva a melhores resultados das abordagens LSH.

Capítulo 6

Conclusões e Trabalhos Futuros

Ao longo deste trabalho, foram exploradas algumas abordagens de *Locality-Sensitive Hashing* para calcular a similaridade entre diferentes documentos textuais através da estimativa do tamanho relativo da sua interseção. Cada uma delas buscava analisar uma forma distinta de representar os documentos de alta dimensionalidade como conjuntos de *tokens*, segundo *operadores* selecionados de acordo com um critério pré-estabelecido. Assim, tendo como modelo o algoritmo *Minwise Hashing*, que utiliza o menor elemento de um conjunto para representá-lo, criamos um método similar, chamado *Maxwise Hashing*. Embora seja análogo ao *Minwise Hashing*, a amostragem realizada pelo *Maxwise Hashing* é através do maior elemento de um conjunto.

Além dessas abordagens, também estudamos outras representações para conjuntos, todas elas usando dois *operadores*. E, desta forma, propusemos os métodos *MinMinwise Hashing*, *MaxMaxwise Hashing* e *MinMaxwise Hashing*. Essas abordagens também buscam analisar as fronteiras de um conjunto para poder identificar a similaridade entre documentos, entretanto, ao sintetizá-los, agregam o dobro de informação produzida pelo *Minwise Hashing* e pelo *Maxwise Hashing*. E assim, era esperado que tais métodos levassem a um ganho linear em qualidade nos resultados de problemas de recuperação de informação.

À medida que foram desenvolvidos os métodos apresentados, procuramos avaliar os seus resultados em um problema real. Para isso, optamos por realizar nossos experimentos utilizando o *corpus* METER, uma base de textos jornalísticos. O *corpus* METER nos permitiu aplicar os métodos propostos na tarefa de identificação de reúso de texto e comparar seus resultados aos gerados pelas abordagens *Minwise Hashing* e *Bag of Words*, uma das representações mais frequentemente usadas em modelos de Recuperação de Informação. Durante as execuções realizadas, constatou-se que as abordagens de *Locality-Sensitive Hashing* tiveram uma perceptível superioridade em relação à abordagem *Bag of Words* para as quantidades de dimensões representativas utilizadas nos experimentos.

Analisando os métodos, identificamos uma semelhança entre os resultados calculados pelo *Minwise Hashing* e pelo *Maxwise Hashing*. Já os testes realizados com as abordagens *MinMinwise Hashing* e pelo *MaxMaxwise Hashing* apresentaram um ganho linear em relação àqueles produzidos pelos métodos que empregam apenas um *operador* em sua função de seleção. Também pudemos constatar que o método *MinMaxwise Hashing* apresentou resultados superiores em relação aos das outras abordagens propostas, podendo ser considerado superlinear.

É importante ressaltar que as famílias de técnicas *Locality-Sensitive Hashing* se propõem a representar conjuntos através de uma versão resumida dos mesmos. Consequentemente, há uma perda de informação nessas versões sintetizadas e, à medida que utilizamos maiores dimensões para representar os conjuntos, esta perda se torna mais evidente. Ou seja, ao dar continuidade aos testes com o aumento de dimensões para representar os conjuntos, métodos como o *Bag of Words* apresentam uma forte tendência de gerar melhores resultados. As técnicas *Locality-Sensitive Hashing*, para menores dimensões, entretanto, se sobressaem, como foi identificado nos experimentos de todos os métodos LSH estudados no presente trabalho.

Como trabalho futuro, sugerimos a aplicação das técnicas aqui estudadas em outras bases reais de Recuperação de Informação. Existem diferentes problemas em que se pode analisar as abordagens propostas, sobretudo aqueles relacionados à identificação de casos de co-derivação e reutilização textual, além da detecção de plágio. Desta forma, poderemos validar a superioridade do método *MinMaxwise Hashing* em relação aos outros.

Além disso, também acreditamos que, uma vez que as abordagens utilizadas nesse trabalho são baseadas no conceito de semi-reticulados e reticulados, se possa dar prosseguimento a esse trabalho através do estudo de outras formas de explorar os limites de um conjunto ordenado. Considerando que também podemos compreender os limites mínimo e máximo de um conjunto como operadores de agregação, sugere-se que sejam ainda analisados outros *operadores*, como por exemplo, a média aritmética ou a mediana.

Referências Bibliográficas

APOSTOLICO, A., BAEZA-YATES, R., MELUCCI, M., 2006, "Advances in Information Retrieval: An Introduction to the Special Issue", *Inf. Syst.*, v. 31, n. 7 (nov.), pp. 569–572. ISSN: 0306-4379. doi: 10.1016/j.is.2005.11.005. Disponível em: <http://dx.doi.org/10.1016/j.is.2005.11.005>.

BAEZA-YATES, R., RIBEIRO-NETO, B., OTHERS, 1999, *Modern information retrieval*, v. 463. ACM press New York.

BARRÓN-CEDEÑO, A., 2010, "On the mono-and cross-language detection of text reuse and plagiarism". In: *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pp. 914–914. ACM.

BARRÓN-CEDEÑO, A., EISELT, A., ROSSO, P., 2009, "A Comparison of Models over Wikipedia Articles Revisions", *ICON*, v. 2009.

BEDREGAL, B. C., SANTOS, H. S., CALLEJAS-BEDREGAL, R., 2006, "T-norms on bounded lattices: t-norm morphisms and operators". In: *Fuzzy Systems, 2006 IEEE International Conference on*, pp. 22–28. IEEE.

BENDERSKY, M., CROFT, W. B., 2009, "Finding text reuse on the web". In: *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pp. 262–271. ACM.

BERNSTEIN, Y., ZOBEL, J., 2006, "Accurate discovery of co-derivative documents via duplicate text detection", *Information Systems*, v. 31, n. 7, pp. 595 – 609. ISSN: 0306-4379. doi: http://dx.doi.org/10.1016/j.is.2005.11.006. Disponível em: <http://www.sciencedirect.com/science/article/pii/S030643790500092X>. (1) {SPIRE} 2004 (2) Multimedia Databases.

BERNSTEIN, Y., ZOBEL, J., 2004, "A scalable system for identifying co-derivative documents". In: *String Processing and Information Retrieval*, pp. 55–67. Springer.

BRIN, S., DAVIS, J., GARCIA-MOLINA, H., 1995, "Copy detection mechanisms for digital documents". In: *ACM SIGMOD Record*, v. 24, pp. 398–409. ACM.

BRODER, A. Z., 1997, "On the resemblance and containment of documents". In: *Compression and Complexity of Sequences 1997. Proceedings*, pp. 21–29. IEEE.

BRODER, A. Z., 2000, "Identifying and filtering near-duplicate documents". In: *Combinatorial pattern matching*, pp. 1–10. Springer.

BRODER, A. Z., CHARIKAR, M., FRIEZE, A. M., et al., 1998, "Min-wise independent permutations". In: *Proceedings of the thirtieth annual ACM symposium on Theory of compu-ting*, pp. 327–336. ACM.

BUHLER, J., 2001, "Efficient large-scale sequence comparison by locality-sensitive hashing", *Bioinformatics*, v. 17, n. 5, pp. 419–428.

BUTTLER, D., 2004, "A short survey of document structure similarity algorithms". In: *International Conference on Internet Computing*, pp. 3–9.

CARDOSO, O. N. P., 2000, "Recuperação de informação", *Lavras,[sd]*.

CARTER, J. L., WEGMAN, M. N., 1977, "Universal classes of hash functions". In: *Proceedings of the ninth annual ACM symposium on Theory of computing*, pp. 106–112. ACM.

CHARIKAR, M. S., 2002, "Similarity estimation techniques from rounding algorithms". In: *Proceedings of the thiry-fourth annual ACM symposium on Theory of com-puting*, pp. 380–388. ACM.

CHUM, O., PHILBIN, J., ZISSERMAN, A., 2008, "Near Duplicate Image Detection: min-Hash and tf-idf Weighting." In: *BMVC*, v. 810, pp. 812–815.

CLOUGH, P., 2010, "Measuring Text Reuse in the News Industry". In: Bently, L., Davis, J., Ginsburg, J. C. (Eds.), *Copyright and Piracy: An Interdisciplinary Critique*, Cambridge University Press.

CLOUGH, P., GAIZAUSKAS, R., PIAO, S. S., et al., 2002a, "Meter: Measuring text reuse". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pp. 152–159. Association for Computational Linguistics, a.

CLOUGH, P., GAIZAUSKAS, R. J., PIAO, S. S., 2002b, "Building and annotating a corpus for the study of journalistic text reuse." In: *LREC 2002*, pp. 1678–1685. European Language Resources Association, b.

CLOUGH, P., OTHERS, 2003, "Old and new challenges in automatic plagiarism detection". In: *National Plagiarism Advisory Service, 2003; http://ir. shef. ac. uk/cloughie/index. html*. Citeseer.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., et al., 2009, *Introduction to Algorithms, Third Edition*. 3rd ed. , The MIT Press. ISBN: 0262033844, 9780262033848.

CORNELIS, C., VERBIEST, N., JENSEN, R., 2010, "Ordered weighted average based fuzzy rough sets". In: *Rough Set and Knowledge Technology*, Springer, pp. 78–85.

CROFT, W. B., METZLER, D., STROHMAN, T., 2010, *Search engines: Information retrieval in practice*. Addison-Wesley Reading.

DE ANDRADE LEITE, M. A., 2009, *Modelo Fuzzy para Recuperação de Informação Utilizando Múltiplas Ontologias Relacionadas*. Tese de Doutorado, Universidade Estadual de Campinas.

DICE, L. R., 1945, "Measures of the amount of ecologic association between species", *Ecology*, v. 26, n. 3, pp. 297–302.

EASTLAKE, D., JONES, P., 2001. "US secure hash algorithm 1 (SHA1)". .

FERNEDA, E., 2003, *Recuperação de informação: análise sobre a contribuição da ciência de computação para a ciência da informação*. Tese de Doutorado.

FODOR, J., MARICHAL, J.-L., ROUBENS, M., 1995, "Characterization of the ordered weighted averaging operators", *Fuzzy Systems, IEEE Transactions on*, v. 3, n. 2, pp. 236–240.

FORMAN, G., ESHGHI, K., CHIOCCHETTI, S., 2005, "Finding similar files in large document repositories". In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pp. 394–400. ACM.

GAIZAUSKAS, R., FOSTER, J., WILKS, Y., et al., 2001, "The METER corpus: a corpus for analysing journalistic text reuse". In: *Proceedings of the Corpus Linguistics 2001 Conference*, pp. 214–223. Citeseer.

GIONIS, A., INDYK, P., MOTWANI, R., et al., 1999, "Similarity search in high dimensions via hashing". In: *VLDB*, v. 99, pp. 518–529.

HAUSBURG, M., RICHTER, R., BRESSLER, I., 2008. "US secure hash algorithm 1 (SHA1)". .

HEINTZE, N., OTHERS, 1996, "Scalable document fingerprinting". In: *1996 USENIX workshop on electronic commerce*, v. 3.

HOAD, T. C., ZOBEL, J., 2003, "Methods for identifying versioned and plagiarized documents", *Journal of the American society for information science and technology*, v. 54, n. 3, pp. 203–215.

INDYK, P., MOTWANI, R., 1998, "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality". In: *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pp. 604–613, New York, NY, USA. ACM. ISBN: 0-89791-962-9. doi: 10.1145/276698.276876. Disponível em: <http://doi.acm.org/10.1145/276698.276876>.

JACCARD, P., 1908, *Nouvelles recherches sur la distribution florale.*

KARP, R. M., 1991, "An introduction to randomized algorithms", *Discrete Applied Mathematics*, v. 34, n. 1–3, pp. 165 – 201. ISSN: 0166-218X. doi: http://dx.doi.org/10.1016/0166-218X(91)90086-C. Disponível em: <http://www.sciencedirect.com/science/article/pii/0166218X9190086C>.

KARP, R. M., RABIN, M. O., 1987, "Efficient Randomized Pattern-matching Algorithms", *IBM J. Res. Dev.*, v. 31, n. 2 (mar.), pp. 249–260. ISSN: 0018-8646. doi: 10.1147/rd.312.0249. Disponível em: <http://dx.doi.org/10.1147/rd.312.0249>.

KULIS, B., GRAUMAN, K., 2009, "Kernelized locality-sensitive hashing for scalable image search". In: *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 2130–2137. IEEE.

LEVY, D. M., 1993, "Document reuse and document systems", *Electronic Publishing*, v. 6, n. 4, pp. 339–348.

LYON, C., MALCOLM, J., DICKERSON, B., 2001, "Detecting short passages of similar text in large document collections". In: *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pp. 118–125.

MANBER, U., OTHERS, 1994, "Finding Similar Files in a Large File System." In: *Usenix Winter*, v. 94, pp. 1–10.

MANNING, C. D., RAGHAVAN, P., SCHÜTZE, H., 2008, *Introduction to information retrieval*, v. 1. Cambridge university press Cambridge.

MARTIN, B., 1994, "Plagiarism: a misplaced emphasis", *Journal of Information Ethics*, v. 3, n. 2, pp. 36–47.

MAURER, H. A., KAPPE, F., ZAKA, B., 2006, "Plagiarism-A Survey." *J. UCS*, v. 12, n. 8, pp. 1050–1084.

MOTWANI, R., RAGHAVAN, P., 1996, "Randomized algorithms", *ACM Computing Surveys (CSUR)*, v. 28, n. 1, pp. 33–37.

MOTWANI, R., RAGHAVAN, P., 2010, "Algorithms and Theory of Computation Handbook". Chapman & Hall/CRC, cap. Randomized Algorithms, pp. 12–12. ISBN: 978-1-58488-822-2. Disponível em: <`http://dl.acm.org/citation.cfm?id=1882757.1882769`>.

NATION, J., 2009, "Notes on lattice theory", *Cambridge studies in advanced mathematics*, v. 60.

NERY, G., BRAGAGLIA, A. P., CLEMENTE, F., et al., 2010, "Nem tudo que parece é plágio: cartilha sobre plágio acadêmico", *Instituto de Arte e Comunicação Social da Universidade Federal Fluminense–UFF, Rio de Janeiro/RJ*.

NEZHAD, A. D., DAVVAZ, B., 2009, "An introduction to the theory of Hv-semilattices", *Bulletin of the Malaysian Mathematical Sciences Society*, v. 32, n. 3, pp. 375–390.

PAGH, R., STÖCKEL, M., WOODRUFF, D. P., 2014, "Is min-wise hashing optimal for summarizing set intersection?" In: *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 109–120. ACM.

PEDRYCZ, W., GOMIDE, F., 1998, *An introduction to fuzzy sets: analysis and design*. Mit Press.

PHILLIPS, J., 2012. "Min Hashing". Disponível em: <`http://goo.gl/zPJPRb`>. [Online; acessado em 02-Janeiro-2015].

PRATT, V., 2015. "Chapter 1 - Lattice theory". Disponível em: <`http://boole.stanford.edu/cs353/handouts/book1.pdf`>.

RAJARAMAN, A., ULLMAN, J. D., 2011, *Mining of massive datasets*. Cambridge University Press.

RIVEST, R., 1992, "The MD5 message-digest algorithm", .

ROMAN, S., 2008, *Lattices and ordered sets*. Springer.

SHIVAKUMAR, N., GARCIA-MOLINA, H., 1996, "Building a scalable and accurate copy detection mechanism". In: *Proceedings of the first ACM international conference on Digital libraries*, pp. 160–168. ACM.

SLANEY, M., CASEY, M., 2008, "Locality-sensitive hashing for finding nearest neighbors [lecture notes]", *Signal Processing Magazine, IEEE*, v. 25, n. 2, pp. 128–131.

STAMATATOS, E., 2011, "Plagiarism detection using stopword n-grams", *Journal of the American Society for Information Science and Technology*, v. 62, n. 12, pp. 2512–2527.

STEIN, B., ZU EISSEN, S. M., 2006, "Near similarity search and plagiarism analysis". In: *From Data and Information Analysis to Knowledge Engineering*, Springer, pp. 430–437.

STINSON, D., 1994, "Combinatorial techniques for universal hashing", *Journal of Computer and System Sciences*, v. 48, n. 2, pp. 337 – 346. ISSN: 0022-0000. doi: http://dx.doi.org/10.1016/S0022-0000(05)80007-8. Disponível em: <`http://www.sciencedirect.com/science/article/pii/S0022000005800078`>.

WIKIPÉDIA, 2013. "Wikipédia: Sobre a Wikipédia". Disponível em: <`http://goo.gl/dOPXEh`>. [Online; acessado em 15-Novembro-2014].

WILKS, Y., 2004, "On the ownership of text", *Computers and the Humanities*, v. 38, n. 2, pp. 115–127.

YAGER, R. R., 1988, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking", *Systems, Man and Cybernetics, IEEE Transactions on*, v. 18, n. 1, pp. 183–190.

ZHANG, H., CHOW, T. W., 2011, "A coarse-to-fine framework to efficiently thwart plagiarism", *Pattern Recognition*, v. 44, n. 2, pp. 471 – 487. ISSN: 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2010.08.023. Disponível em: <`http://www.sciencedirect.com/science/article/pii/S0031320310004097`>.

Apêndice A

Abordagens baseadas em Recuperação da Informação para detecção de reúso de texto

Ao longo do estágio desenvolvido no Institut de Recherche en Informatique de Toulouse, foi realizado um trabalho com objetivo de estudar o desempenho de métodos clássicos de Recuperação da Informação aplicados ao problema de detecção de reutilização de texto. A seguir, apresentamos o relatório produzido a partir deste estudo.

Information Retrieval based approaches for text reuse detection

Internship Report
Institut de Recherche en Informatique
de Toulouse

Danielle Caled Vieira

December 2015

1 Introduction

Problems in Information Retrieval area, such as the detection of text reuse, are a subject of considerable theoretical and practical interest that have been widely studied and discussed [Clough et al., 2002a]. Sometimes, when performing an Information Retrieval task on a large set of texts, it is necessary to search for any element that might indicate a suspected case [Barrón-Cedeño, 2010]. One of the main ways to solve this problem is by measuring the similarity between documents [Barrón-Cedeño, 2010]. In this work, we intend to apply different techniques to perform the identification of reuse in journalistic texts.

It is not a novelty that media enterprises usually sign for the service of news agencies and use the content produced by them with distinct levels of changes. These news agency, however, normally do not have information of the portion of their texts that is reused by common media. For instance, they do not evaluate whether their texts are wholly or partially reused by their media customers or if they are the only source used on the material produced by their customers. Given this context, it is an important task to study the use of the material produced by news agencies. Therefore, the information obtained from it could be used in many situations, like:

- News agency could follow the rates in which their media customers are using their material, and, so, they could predict the influence and relevance they have on their clients material.
- News agencies could evaluate which domains are more interesting to their customers and which ones are not so relevant to them, and thus, they could better decide whether or not investing their effort in some subjects.

- News agency could compare the costumer's usage of their material with the usage of the material generated by their competitors. After that, they could better understand the relation between their clients and their competitors and they could create a strategy to defeat their rivals in a market segment.
- A media customer which signs for the services offered by different news agencies could perform a periodical evaluation of its needs. And, therefore, reduce their expenses, hiring only the services that are really used on their material.
- A media customer could analyse whether invest and allocate the resources and efforts of its team, taking in consideration the usage of the hired services.

For the study of journalistic reuse, we performed different experiments on a corpus created by the Departments of Journalism and Computer Science at Sheffield University. This corpus, METER corpus, consists of a set of texts produced by the biggest British news agency (Press Association - PA) and a set of stories about the same events published in nine British newspapers.

We conducted our experiments using different classical Information Retrieval models to attempt to identify journalistic reuse on METER corpus. According to Zhai [2008], Vector Space Model, Okapi BM25 and Language Model are very effective techniques. Therefore, the following techniques were used during our experiments: Vector Space Model with TF-IDF, Language Model and Okapi BM25.

In the following sections we briefly explain Text Reuse, and, more specifically, Journalistic Reuse and we also describe the corpus we use on this work (Section 2). In Section 3, we list different Information Retrieval Models (Boolean Model, Vector Space Model, Okapi BM25 and Language Model). We talk about our experiments and tools, showing our results in Section 4. We discuss our results in Section 5 and, finally, we present the conclusion of this work in Section 6.

2 Text Reuse

The reuse of texts, concepts and content is a very explored and often studied production method [Wilks, 2004, Levy, 1993, Clough, 2010, Barrón-Cedeño, 2010, Clough et al., 2002b, Bendersky and Croft, 2009, Clough et al., 2002a]. Present both in old narratives as in written documents, the reuse has increased considerably with the use of digital technology as writing tools [Wilks, 2004, Levy, 1993, Clough, 2010]. Other advances in information technology, such as Internet access and the use of search engines, also contribute to the occurrence of different types of text reuse [Clough, 2010]. Some examples of reuse include the creation of literary and historical texts, translation, review and summary of existing texts [Clough et al., 2003].

Clough et al. [2003] define reuse text as "the activity whereby pre-existing written material is reused during the creation of a new text, either intentionally or un-intentionally." Text reuse involves a process in which an author rewrites or edits, with or without permission from the owner, a given document [Clough, 2010]. Because of the difficulty encountered in developing a new idea, it is very likely that this author bases his studies on previous related work [Barrón-Cedeño, 2010].

According to Clough et al. [2002b], it is important to realize that the reuse of text is a continuous phenomenon that extends from literal word-for-word reuse (*verbatim*) to "through varying degrees of transformation involving substitutions, insertions, deletions and reorderings, to a situation where the text has been generated completely independently, but where the same events are being described by another member of the same linguistic and cultural community (and hence where one can anticipate overlap of various sorts)".

The expression "text reuse" covers a range of textual transformations, such as summaries, exact reproductions, reformulations from other sources and reports (which have little in common with the original document, except the subject) [Bendersky and Croft, 2009]. The transformations range from additions of new facts, events or opinions to the source text, deletion of original parts, subtle changes in writing, complete reformulation of the text, changes in the style of writing to meet a different demand, simplification, translation of an original text to another language, among others manipulation processes [Clough et al., 2003, Bendersky and Croft, 2009, Barrón-Cedeño, 2010].

Clough [2010] deals with the text reuse problem from two perspectives, the author's and the reader's. The first, the author's perspective, is to search and edit the material of interest [Levy, 1993, Clough, 2010]. The second, the reader's perspective, is seen as an analytical problem of authorship attribution in which, considering two texts, the aim is to find out if one of them is derived from the other [Wilks, 2004, Clough, 2010].

According Wilks [2004], reuse is an independent form of linguistic activity and the computational methods used to detect it subtly differ from those used in plagiarism problems [1]. Identifying text reuse can become a complicated task because of the many changes that may occur in the text, such as literal reuse, or more complex cases, like paraphrase and summarization, that make the new version very different from the original [Clough, 2010]. For this reason, it is quite obvious the subjectivity in evaluating reuse detection [Clough et al., 2002b].

In the context of text reuse, it is important to define the relationships between documents. According to each type of relationship, it is possible to choose an appropriate approach that may indicate reuse. Nevertheless, the exact search for fragments of documents is generally not satisfactory, although it is an easy approach to be used [Heintze et al., 1996]. Once there are more elaborate types of reuse, this approach fails even to capture fragments that have been slightly modified, ignoring most of interesting textual relationships [Heintze et al., 1996].

[1] Plagiarism is characterized when using "ideas, concepts or phrases of another author (who formulated and published them), without giving him credit, without citing it as a source of research." [NERY et al., 2010]

Heintze et al. [1996], in their work, consider significant the following types of relationship among documents:

(a) Identical documents.

(b) Documents generated by minor editions/corrections on another document(s).

(c) Documents generated from reorganization of the content of other documents.

(d) Documents that are revisions of other documents.

(e) Documents generated by condensation/expansion of other documents.

(f) Documents comprising portions of text provenient from other documents.

Barrón-Cedeño [2010], moreover, enumerates a list of situations where text reuse occurs. It was based on the work Martin [1994] and Maurer et al. [2006]. Both authors identify several methods used in plagiarism, but Barrón-Cedeño [2010] selected some that he recognizes actually as text reuse methods. Next, we explain the situations in which text reuse is more frequent:

(a) Word-for-word reuse (*verbatim*): It is the reuse most obvious and likely type of reuse [Martin, 1994]. It occurs when someone copies unchanged fragments of a published work without using quotation marks and/or referencing the source [Martin, 1994, Barrón-Cedeño, 2010]. This type of reuse is the classic "copy and paste" [Maurer et al., 2006].

(b) Paraphrase: It occurs when a textual content is copied, even with minor modifications. Few words of the original text can be changed or the text can be modified or paraphrased [Martin, 1994, Barrón-Cedeño, 2010].

(c) Reuse of ideas: It occurs when, although there is no dependence in terms of words or the source's form, an original idea is reused [Martin, 1994].

(d) Reuse of source code: If happens with the use of programming code, algorithms, classes, or functions without proper citation of the author or his permission [Maurer et al., 2006, Barrón-Cedeño, 2010].

(e) Translated reuse: It happens when there is a translation of some content, even with some modifications, without the reference to the original text [Maurer et al., 2006, Barrón-Cedeño, 2010].

There are situations where cases of reuse are well accepted and considered appropriate. They are seen as benign cases of adaptation, rewriting or plagiarism, made in situations where the author of the original document is not deceived [Wilks, 2004]. An example in the journalistic field is the information generated by news agencies [2] that is sold to media customers, such as newspapers, magazines, radio and other types media to their own use [Clough et al.,

[2]The best-known news agencies are the Press Association and Reuters (both in the UK) and the Associated Press and UPI (US) [Wilks, 2004].

2002a, Wilks, 2004]. In exchange for a fee paid to news agencies, the interested media have access to their service and can use the content as they wish (that is, publishing the story *verbatim* or rewriting the text so it is compatible with their style and/or interest) [Barrón-Cedeño, 2010].

Clough et al. [2002b] state that the process of editing and publishing news in newspapers is a complex and specialized task, and on several occasions, journalists are forced to resort to wire services as source of their stories. Usually, items prepared by newspapers are conditioned to some restrictions, such as:

- Short deadlines;
- Prescriptive practices of writing;
- Physical size limitation in the layout;
- Readability and understanding of the target audience;
- Editorial bias;
- Newspaper style.

The following example, taken from [Clough et al., 2002b], illustrates the types of journalistic rewrites performed by newspapers on a short sentence written by PA. Although conserving the same content of the source, the rewrites are aligned to the style and form of the newspaper that published them.

Original (PA): A drink-driver who ran into the Queen Mother's official Daimler was fined £700 and banned from driving for two years.
Rewrite (The Sun): A DRUNK driver who ploughed into the Queen Mother's limo was fined £700 and banned for two years yesterday.
Rewrite (The Mirror): A BOOZY driver who smashed into the Queen Mums's chauffer-driven Daimler minutes after she had been dropped off was banned for two years and fined £700 yesterday.
Rewrite (Daily Star): A DRUNK driver who crashed into the back of the Queen Mum's limo was banned for two years yesterday.

2.1 *Corpus*

In this work, we will perform different experiments to identify journalistic text reuse. Therefore, we chose a collection of journalistic stories called the METER[3] (Measuring TExt Reuse) *corpus*. The creation of this dataset is minutely described by [Clough et al., 2002b, Gaizauskas et al., 2001].

The METER *corpus* [Clough et al., 2002b] was developed by the Departments of Journalism and Computer Science at Sheffield University. The project also had the collaboration of Press Association [4] (PA), the major news agency

[3] http://nlp.shef.ac.uk/meter/
[4] http://www.pressassociation.com/

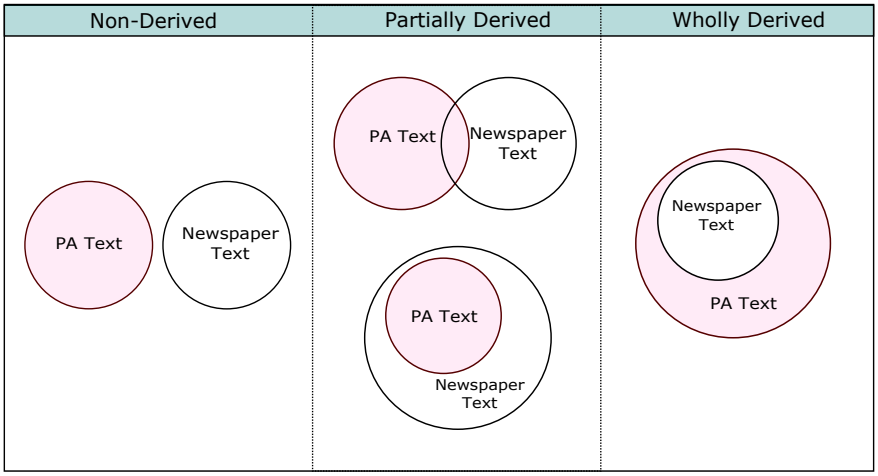

Figure 1: Types of relations between PA texts and a newspaper text. Adapted from [Clough, 2001].

from the United Kingdom, whose stories are frequently used directly or indirectly by British newspapers.

The *corpus* comprises two sets of stories. The first set is composed of stories written by PA, while the other set consists of stories about the same events published by nine British newspapers (The Sun, Daily Mirror, Daily Star, Daily Mail, Daily Express, The Times, The Daily Telegraph, The Guardian and The Independent). Although not using directly the stories written by PA, theses newspapers may refer to the service provided by the Press Association during the production of their reports [Gaizauskas et al., 2001]. Thus, in a collection of news written by agencies (such as PA) and texts produced by newspapers on the same topic, different examples are expected to be "real" cases of text reuse [Gaizauskas et al., 2001].

For the creation of METER *corpus*, 1,716 texts (over 500,000 words) were collected between the months of July 1999 and June 2000. Among the texts which compose the *corpus*, 944 are newspaper articles and 772 are news produced by the PA. In order to represent the dependence of the contents of each journalistic text regarding the PA texts, each document was categorized by a trained journalist as wholly, partially and non-derived [Gaizauskas et al., 2001] (Figure 1).

(a) **Wholly derived:** All content of target text was derived from the PA articles. That is, all its facts can be mapped to one or more PA texts. The texts may be copied in whole or modified in various ways, including reordering of words, replacement of some terms by synonyms and paraphrases.

(b) **Partially derived:** Part of the target text content is derived from the

news published by PA, although other sources may also have been used. In partially derived texts, only some sections can be mapped to their correspondent PA sources.

(c) **Non-derived:** No content of target text is derived from the news published by PA, although some related words can be common to both texts. This class includes newspaper publications that are written independently of PA items. Even dealing with the same issues that some texts of the PA, the articles in this category not used as a source.

METER *corpus* contains 300 wholly-derived articles , 438 partially-derived articles and 206 non-derived articles. That is, approximately 78.2% of the documents published by newspapers are derived from news written by the PA.

3 Information Retrieval

According to the definition of Manning et al. [2008], in an academic context, Information Retrieval (IR) is "the action of finding material (usually documents) of unstructured nature (usually text) in large collections (usually stored on computers) that satisfies a demand for information". To Baeza-Yates et al. [1999], the Information Retrieval "deals with the representation, storage, organization of, and access to information items".

In order to better understand the IR, it is also necessary to define other concepts, namely, "A document is a data object, usually textual, though it may also contain other types of data such as photographs, graphs, and so on" [Baeza-Yates and Frakes, 1992]. Manning et al. [2008] consider "whatever units we have decided to build a retrieval system over" a document. A group of documents used in the information retrieval process is called a collection of documents or corpus (*corpora*, in plural) [Manning et al., 2008]. Documents of a collection are often represented as a set of terms or keywords [Baeza-Yates et al., 1999]. Terms are indexing units, usually words, directly extracted from the document or specified by experts, which constitute a logical view of the document [Baeza-Yates et al., 1999, Manning et al., 2008].

3.1 Conceptual models of Information Retrieval

One of the main goals of academic research in Information Retrieval area has been to understand and formalize the steps that constitutes the decision-making process by a human agent, that decides if a fragment of the text is relevant to his demand [Croft et al., 2010]. As we are not able to reproduce the representation and language processing by the human brain yet, we deal with IR procedures by proposing theories in the form of mathematical models of recovery and testing these theories by comparing them to human actions [Croft et al., 2010].

Thus, good models must produce results that are correlated with human decisions about the relevance of documents [Croft et al., 2010]. Generally, a relevant document is recognized by users as the one that contains the information

that a person was looking for to submit a query in a search engine [Manning et al., 2008, Croft et al., 2010]. However, as the concept of relevance is a subjective definition, two people may disagree about the relevance of a document to a query and it is extremely complex to justify why a document is more relevant than another [Croft et al., 2010].

A conceptual model of IR is a generic approach for retrieval systems [Baeza-Yates and Frakes, 1992]. Baeza-Yates et al. [1999] propose, in their book, a taxonomy to categorize 15 models of information retrieval. Like all mathematical models, the IR models provide a framework that enables defining new tasks and understanding the results [Croft et al., 2010].

According to Baeza-Yates et al. [1999], to build a model, you must first determine the representations of documents and the demands of users. Then you must define the framework in which such representations can be modeled. It is important that the framework is able to provide means for construction of a ranking function. Baeza-Yates et al. [1999] still exemplify the use of framework considering the IR classic models: in the Boolean model, the framework consists of sets of documents and standard operations between sets; in the vector model, the framework consists of a n-dimensional vector space and linear algebra operations on vectors; while for the probabilistic model, the framework is comprised of sets, probability operations and Bayes' theorem.

For a better understanding of Information Retrieval models that will be presented here, it is important to already introduce some definitions. In these models, documents are interpreted as sets of terms. Some terms of a document are more relevant than others, and may be representative in the semantic context of the document. These are called index terms, and are used to index and summarize the contents of documents [Baeza-Yates et al., 1999]. As the relevance of each term in the description of documents content is a concept variable and depends of the analyzed corpus, it is necessary to use a mechanism to capture this feature of indexing terms. This is done through the use of numerical weights for each index term in a document [Baeza-Yates et al., 1999].

According to Baeza-Yates et al. [1999], the employment of weights aims to quantify the importance of the index term to describe the document content. Therefore, for a tuple $[t_i, d]$, where t is a generic index term and d is a document, the weight is given by $d_i \geq 0$. For an index term that does not appear in the document, the value of d_i is 0. Thus, the document d is represented by the vector $\vec{d} = \{d_1, d_2, \cdots, d_n\}$, where n is the number of index terms of the system. The function that returns the weight associated with the term t_i in a n-dimensional vector is given by g_i, where $g_i(\vec{d}) = d_i$.

3.1.1 Boolean Model

The Boolean retrieval model is one of the oldest and simplest search engines. This model uses a framework based on the set theory and Boolean algebra. Queries are specified as Boolean expressions with precise semantics and only the documents containing terms that match the logical expression used in the query are recovered [Baeza-Yates et al., 1999, Cardoso, 2000]. Because of its

simplicity and clear formalism, this model has been widely adopted [Baeza-Yates et al., 1999].

The Boolean model gets its name in reference to the Boolean algebra, whose formalism was introduced by George Boole in 1854. As in Boolean algebra, this model works with sets and logical operators. In the Boolean model, the document is seen as a set of terms. Furthermore, the representation of any query is made in the form of a Boolean expression of its terms, that is, the terms are combined with Boolean logical operators (`AND, OR, NOT`) [Manning et al., 2008]. This model, as well as Boolean algebra, has only two possible results, the values `TRUE` and `FALSE` [Croft et al., 2010].

With regard to the relationship between terms and documents in Boolean model, there are two possibilities: a term to be whether or not contained in the document. That is, it is a binary relationship in which the weight d_i assumed by index term t_i in the document d can be represented as $d_i \in \{0, 1\}$ [Baeza-Yates et al., 1999].

Baeza-Yates et al. [1999] formally define the Boolean model considering a query q as a conventional Boolean expression; $\vec{q}_{dnf}$, the disjunctive normal form for the query q; and let $\vec{q}_{cc}$ be any of the conjunctive components of $\vec{q}_{dnf}$, the similarity $sim(d, q)$ of a document d for the query q is defined by 1 [Baeza-Yates et al., 1999].

$$sim(d,q) = \begin{cases} 1 & if\ \exists \vec{q}_{cc}\ \mid\ (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall_{t_i},\ g_i(\vec{d}) = g_i(\vec{q}_{cc})) \\ 0 & otherwise \end{cases} \tag{1}$$

If the similarity $sim(d, q)$ is 1, the model predicts that the document d is relevant to the query q. Otherwise, the document will not be considered relevant.

Since it assumes that all documents in the retrieved set are equivalent in terms of relevance (because relevance is treated in binary form), the Boolean model is not considered a ranking algorithm [Croft et al., 2010].

As explained above, the inability of the Boolean model to attribute different degrees of importance to each document consists of a restriction on its use and can make it difficult to obtain good results using this model. In addition to this limitation, some Boolean operations can be very complex because the expressions used in the search, that must be accurate semantically [Baeza-Yates et al., 1999]. More complex expressions require the users to have a solid knowledge in Boolean logic [Ferneda, 2003]. Finally, another difficulty encountered in using the Boolean model is the little control over the amount of documents that are returned in a search [Ferneda, 2003]. This can lead the user to perform various interactions, seeking the logical refinement of Boolean expression closest to the ideal, that returns the desired quantity of documents.

Whilst providing these limitations, the Boolean model is still a widely used model, being present in many Information Retrieval systems [Croft et al., 2010, Ferneda, 2003]. Baeza-Yates et al. [1999] indicated that the main advantages of the model Boolean are the "clear formalism behind the model and its simplicity". Nevertheless, the model results are easy to predict and understanding. We can use any feature of documents (word, document date, document type, etc.) as

an operand of a Boolean query [Croft et al., 2010]. From the point of view of implementation, since they do not need to perform the sorting of results, the Boolean model is generally more efficient than models that rank documents [Croft et al., 2010].

3.1.2 Vector Space Model

Vector Space Model (VSM) is repeatedly being adopted in Information Retrieval literature since the 1960s [Croft et al., 2010]. This approach recognizes the limitations of Boolean model to employ binary weights to index terms in queries and documents [Baeza-Yates et al., 1999]. To avoid these restrictions, VSM proposes a framework in which it is possible to obtain documents that respond partially to a search expression [Baeza-Yates et al., 1999, Ferneda, 2003].

In vector model, we attribute non-binary values of weights to index and query terms with the purpose of representing the degree of similarity between each document in the corpus and in the search expression chosen by the user [Baeza-Yates et al., 1999, Ferneda, 2003]. Next, the inverse ordering (decreasing order) is performed on documents, considering their respective degrees of similarity. It is precisely this calculation followed by the ordering that allows the user to capture partial match of the documents in relation to the query. The most significant effect of this process is that, in response to the consultation carried out, the documents ranked set will be much more accurate than that found by the Boolean model [Baeza-Yates et al., 1999].

In this model, both documents and the query are seen as n-dimensional vectors, in which n is the number of index terms [Croft et al., 2010]. A document d is represented by the vector $\vec{d} = (d_1, d_2, \ldots, d_n)$, where the weight d_i assumes a positive and non-binary value. Similarly, a query q is given by the vector $\vec{q} = (q_1, q_2, \ldots, q_n)$.

By understanding the documents and queries as vectors, it is common to see them in the form of vector diagrams. Typically, they are shown as points or vectors in n-dimensional space, in which the n terms represent all the characteristics present in the indexed documents [Croft et al., 2010].

Through this representation, documents can be ranked by calculating the distance between points given by their corresponding vectors and the vector of the query [Croft et al., 2010]. Vector model aims at evaluating the degree of similarity between a document d with respect to the query q as a correlation between the two vectors $\vec{d}$ and $\vec{q}$, so that documents with the highest score are the most similar to the query [Baeza-Yates et al., 1999, Croft et al., 2010]. Different similarity measures are used to quantify the similarity between the two vectors, being the most recurring and consolidated of them, the measure of cosine correlation [Baeza-Yates et al., 1999, Croft et al., 2010, Manning et al., 2008].

The main advantage of the cosine measure is to compensate the effect of the document size by normalizing the document and the query vectors [Manning et al., 2008]. This metric measures the angle between the two vectors, considering that both have the same size, so the cosine value will always be between 0

(in the case of completely distinct vector representations) and 1 (in the case of identical vectors). According Baeza-Yates et al. [1999], the similarity value may be expressed by the cosine of the angle between two vectors using the Equation 2.

$$sim(d,q) = \frac{\vec{d} \cdot \vec{q}}{|\vec{d}| \times |\vec{q}|} = \frac{\sum_{i=1}^{n} d_i \times q_i}{\sqrt{\sum_{i=1}^{n} d_i^2} \times \sqrt{\sum_{i=1}^{n} q_i^2}} \tag{2}$$

where the numerator is sum of the dot product of document d and query q weights, while $|\vec{d}|$ and $|\vec{q}|$ are the norms of the vectors $\vec{d}$ and $\vec{q}$, respectively.

The concept of relevance in the vector model is more flexible than in Boolean model. A user can set a threshold for the similarity, so that all recovered documents d have *sim(d, q)* greater than the threshold [Baeza-Yates et al., 1999]. In this case, documents that meet this restriction are considered relevant to the query q.

The main difference between vector model and Boolean model, as we have seen, is the way weights of terms in the corpus are generated. Several weights allocation schemes have been used over the past few years, however, most of them are variations of the measure *TF-IDF* [Croft et al., 2010]. This measure has two components: the frequency of terms (*Term Frequency* or *tf*) and the inverse of the frequency of a term among documents of the corpus (*Inverse Document Frequency* or *idf*) [Baeza-Yates et al., 1999].

The term frequency measure (*tf*) "reflects the importance of a term in a document" d or in a search expression [Croft et al., 2010]. The *tf* component (3) is defined as the number of times a particular term t appears in a document d [Ferneda, 2003]. That is, the higher the value of *tf*, more important is the term to describe the document.

$$tf(t,d) = count(t,d) \tag{3}$$

The factor *idf* (Inverse Document Frequency) represents the distribution of a term in the corpus. Once frequent terms in documents may not be relevant to the query, the component *idf* assigns a higher weight to terms that appear in fewer documents [Croft et al., 2010]. The computation of *idf* of a term t is given by Equation 4.

$$idf(t) = \log\left(\frac{N}{df}\right) \tag{4}$$

where N is the number of documents in the corpus and *df* is the number of documents containing the term *t*.

Thus, since we defined the *tf* and *idf* components, we can calculate the composite weight *tf-idf* value of each term t in document d through Equation 5.

$$tf\text{-}idf(t,d) = tf(t,d) \times idf(t) \tag{5}$$

According to Manning et al. [2008], the main characteristics of *tf-idf*(t,d) measure are:

(a) Assignment of higher weight value when term t appears many times within a small number of documents;

(b) Assignment of lower weight value when term t appears few times within a document, or appears in many documents;

(c) Assignment of the lowest weight value when the term appears in all documents.

Baeza-Yates et al. [1999] make a critical analysis of the vector model, citing its advantages such as: partial match strategy allowing the retrieval of documents that approximately match the query; similarity ranking by the cosine of the angle between the document and the query vectors and; the use of weight assignment scheme to terms, which improves the performance of IR model.

However, in theory, this model has the limitation of assuming that index terms are mutually independent [Baeza-Yates et al., 1999]. *TF-IDF* measure does not consider the dependencies that may exist between terms of the corpus [Baeza-Yates et al., 1999]. However, "there is no conclusive evidence which point that such dependencies significantly affect the performance of an Information Retrieval system" [Ferneda, 2003].

3.1.3 Okapi BM25

The retrieval function Okapi BM25 has been widely used by search engines and IR researchers to term-weighting and document-scoring function [Pérez-Iglesias et al., 2009, Robertson and Zaragoza, 2009]. BM25 is "one of the most robust and effective retrieval functions" and it is derived from 2-Poisson probabilistic retrieval model [Robertson and Walker, 1994], with some approximations [Robertson et al., 2004].

The BM25 classifies the terms of a document into elite and non-elite [Robertson et al., 2004]. "Eliteness" is a property assigned to each document-term pair that can be understood as a form of aboutness: if the term is elite in a document, the document's subject is likely to be related to the concept denoted by the term [Robertson and Zaragoza, 2009]. Therefore, we can assume that a term occurrence in a document depends on "eliteness" [Robertson and Zaragoza, 2009].

Robertson and Zaragoza [2009] assume that there is an association between "eliteness" e and relevance to a query that may contain many concepts. Term frequency, however, is related only with "eliteness" and it is assumed that the property "eliteness" for each different term in a document is independent [Robertson and Walker, 1994].

BM25 formula is shown in Equation 7, according to the notation defined in [Robertson et al., 2004]. Considering idf' (Equation 6) a close approximation to classical idf [Robertson and Zaragoza, 2009], we can computate the BM25 score as given by Equation 7.

$$idf'(t) = log\left(\frac{N - df(t) + 0.5}{df(t) + 0.5}\right) \tag{6}$$

where t is the term, df is the number of documents containing t and N is the number of documents in the collection.

$$score(d,q) = \sum_{t \in q} idf(t) \cdot \frac{(k_1+1) \cdot tf(t,d)}{tf(t,d) + k_1 \cdot ((1-b) + b \cdot \frac{|d|}{avgdl})} \tag{7}$$

where *tf(t,d)* is the term frequency of t in d, $|d|$ is the document d length and *avgdl* is the document average length along the collection. k_1 and b are free parameters, usually, $k_1 \in [1.2, 2.0]$ and $b = 0.75$.

The parameter k_1 represents how the term weight component changes according to $tf(t,d)$: If $k_1 = 0$, the term frequency would be reduced to a binary element, indicating only the presence or absence of the term [Croft et al., 2010]. If k_1 is large, then, the term weight component would behave almost linearly with term frequency [Croft et al., 2010].

The parameter b represents the impact of the length normalization, that is, when $b = 0$, no length normalization is performed, and when $b = 1$, it means full normalization [Croft et al., 2010].

3.2 Language Model with Dirichlet Similarity

Language Model is defined by Zhai [2008] as a probability distribution over word sequences. It has its foundations in statistical theory, giving any sequence of words a potentially different probability [Zhai and Lafferty, 2001, Zhai, 2008].

Language Model (LM) is based on the idea of estimating a language model for each document, and then, ranking these documents according to the likelihood between their language model and the query [Zhai and Lafferty, 2001]. The LM approach models the idea that "a document is a good match to a query if the document model is likely to generate the query", which is the case of the document and the query sharing many common words [Manning et al., 2008]. This approach, thus, provides a different realization of some of the basic ideas for document ranking

Considering a query q and a document d, the probability of q as being "generated" by a probabilistic model based on d is $p(q|d)$ [Zhai and Lafferty, 2001]. Thus, to discover the documents which could have generated q, we should estimate the posterior probability $p(d|q)$, given by Equation 8, calculated from Bayes' formula [Zhai and Lafferty, 2001].

$$p(d|q) \propto p(q|d) \cdot p(d) \tag{8}$$

In their work, Zhai and Lafferty [2001] intended to estimate a unigram language model based on a given document d. This unigram model (Equation 9) just reflects the frequency of terms in the text and represents the maximum likelihood estimation for the unsmoothed model [Chen and Goodman, 1998, Zhai and Lafferty, 2001].

$$p_{ml}(t|d) = \frac{tf(t,d)}{\sum_{t' \in V} tf(t',d)} \tag{9}$$

where *tf(t,d)* is the term frequency of t in d and V is the set of all terms in the vocabulary.

This unsmoothed model, however, will ignore the probability of an unseen term in the document, and then, underestimate it. So, a smoothing technique aims at improving the accuracy of a term probability estimation by assigning a non-zero probability to the unseen terms [Zhai and Lafferty, 2001]. As the authors say, smoothing methods reduce the probability of terms that are seen in the text and increase the probability of an unseen word according to some "fallback" model. Zhai and Lafferty [2001], following [Chen and Goodman, 1998], present 10 as the general form of a smoothed model.

$$p(t,d) = \begin{cases} p_s(t|d) & \text{if the term } t \text{ is seen} \\ \alpha_d \cdot p(t|C) & \text{otherwise} \end{cases} \tag{10}$$

where the smoothed probability of a term t seen in the document is $p_s(t|d)$, the collection language model is $p(t|C)$ and α_d is a coefficient used for controlling the weight assigned to unseen terms, so that the sum of all probabilities is 1. The coefficient α_d usually depends on the document d, and, for a given value of $p_s(t|d)$, α_d can be calculated from Equation 11 [Zhai and Lafferty, 2001].

$$\alpha_d = \frac{1 - \sum_{t \in V : c(t,d) > 0} p_s(t|d)}{1 - \sum_{t \in V : c(t,d) > 0} p(t|C)} \tag{11}$$

Thus, for Bayesian smoothing using Dirichlet priors, the chosen parameters are given by Equation 12 and equation 13. This language model is a multinomial distribution which has as prior for Bayesian analysis the Dirichlet distribution [Zhai and Lafferty, 2001].

$$p_s(t|d) = \frac{c(t,d) + \mu \cdot p(t|C)}{|d| + \mu} \tag{12}$$

$$\alpha_d = \frac{\mu}{|d| + \mu} \tag{13}$$

4 Experiments

4.1 Document Representation

Before running our experiments, we converted each story produced by PA and the articles published by newspapers from METER corpus in sets of words. This representation is known as Bag of Words [Silva and Ribeiro, 2003]. Thus, each document is indexed according to the set of terms occurring in it, and then, a vector with one entry for each term of the corpus is created [Silva and Ribeiro, 2003]. These entries should be filled with the number of occurrences of the term in the document [Silva and Ribeiro, 2003].

Another decision we had to make during our experiments was related to stop words removal. Along with METER corpus, the authors also provide a stop list. According to [Martin and Jurafsky, 2000], a stop list is "a list of high frequency

words that are eliminated from the representation of documents". The words in this kind of list are called stop words. [Barrón-Cedeño, 2010] lists two reasons for which stop words are generally removed from collections:

(a) Because of their little semantic weight;

(b) To save space while representing documents. This case is justified due to the considerable space the representation of these words demands. Therefore, discarding stop words can reduce by half the size of the corpus.

For these reasons, we opted for executing the experiments both with and without stop words removal, and then, comparing the obtained results.

4.2 Lucene API

To perform the experiments, we used Apache Lucene API [5], a text search engine library written in Java. Lucene provides many methods that allow users to implement Information Retrieval experiments. For this reason, we chose this tool to run our experiments.

Lucene handles some basic Information Retrieval concepts and, sometimes, uses variations of their corresponding formulas, such as in the case of *TF-IDF*, given by Equation 14 and Equation 15.

$$tf(t,d) = count(t,d)^{1/2} \tag{14}$$

$$idf(t) = 1 + log\left(\frac{N}{df(t)+1}\right) \tag{15}$$

where, t is a term, d is a document, N is the sum of documents in the corpus and df is the sum of documents in which t appears.

4.2.1 Vector Space Model

Lucene's implementation of Vector Space Model is combined with an Boolean Model of Information Retrieval. First, documents are "approved" with Boolean Model, and then, they are ranked according to VSM.

Lucene represents documents and queries as weighted vectors in a multi-dimensional space, where each distinct term is a dimension, and the corresponding weights are TF-IDF values. The decision of using TF-IDF as weights in Lucene is due to the belief they produce search results of high quality.

The conceptual scoring formula used by Lucene is shown in Equation 16.

$$score(d,q) = coord(q,d) \cdot qBoost(q) \cdot \frac{\vec{d} \cdot \vec{q}}{|\vec{q}|} \cdot lenNorm(d) \cdot dBoost(d) \tag{16}$$

[5] https://lucene.apache.org/

coord(q,d): It is a score factor based on how many query terms the document contains. Through this factor, users can reward documents matching more query terms.

$\frac{\vec{d}\cdot\vec{q}}{|\vec{q}|}\cdot$ **lenNorm(d)**: Normalizing the cosine through $\mid \vec{d} \mid$ removes all document length information and this could be problematic in some cases. Therefore, Lucene implements a different document length normalization, which normalizes the cosine to a vector equal to or larger than the unit vector and allow the comparison of scores generated from different queries.

qBoost(q): It is a search time boost users can specify for a term in a query text. The contribution of a query term to the score of a document is computed from the product of the boost of that query term.

dBoost(d): At the indexing phase, users can assign a document boost to specify that some documents are important. Then, while computing scores of each document, they are multiplied by their corresponding boost value.

4.2.2 BM25 Similarity

BM25 was integrated into Lucene following the guidelines of Robertson et al. [1995]. The datails of the implementation are better described by Pérez-Iglesias et al. [2009].

4.2.3 Language Model with Dirichlet Similarity

The main difference between Lucene's implementation and Zhai and Lafferty [2001]'s formula is that the authors define a negative score to documents containing the term, but with fewer occurrences than predicted by the collection language model. Lucene's implementation, by the other side, returns 0 for this kind of documents.

4.3 Experiment Configuration and Results

In this work, we studied the approaches cited before (Vector Space Model, BM25 and Language Model) using the framework Lucene. We used different values for the threshold according to experiment results. We also evaluated the models for different values of cut off (the top 4, 5, 10 and the full rank).

For Vector Space Model in Lucene implementation, only the aforementioned values were studied. For, BM25, we also investigated the results for $b = 0.75$ and for $k_1 \in \{1.2, 1.5, 1.7, 2.0\}$. And for Language Model, we also studied the results for $\mu \in \{10, 500, 2000\}$.

After running our experiments, we evaluated the results of our approaches by performing the Cross Validation of each method for the cases of removing and not removing stop words. Thus, we used the holdout method, also called test sample estimation, which consists in dividing the data in two mutually exclusive subsets, one for training and tuning the parameters and, the other for testing them, called holdout set [Kohavi et al., 1995]. To perform Cross-Validation using the holdout method, first, we fit a function using only the training set

and, then, we predict the output for data in the holdout set [Mago, 2011]. The eventual errors generated during the computation are accumulated to give the mean absolute test set error [Mago, 2011].

Manning et al. [2008] affirms that the two most frequent and basic measures used to evaluate Information Retrieval effectiveness are precision and recall. They are defined for the case where we search documents that best fit a query [Manning et al., 2008].

- Precision (Equation 17) represents the number of relevant retrieved documents over all retrieved items [Manning et al., 2008].

$$precision = \frac{\#(relevant\ items\ retrieved)}{\#(retrieved\ items)} \tag{17}$$

- Recall (Equation 18) indicates the fraction of relevant retrieved documents while considering all relevant documents in the system [Manning et al., 2008].

$$recall = \frac{\#(relevant\ items\ retrieved)}{\#(relevant\ items)} \tag{18}$$

A single measure based both on precision and recall is the *F measure* (Equation 19) [Manning et al., 2008]. This measure is used for evaluating effectiveness in some search applications and it is defined as the harmonic mean of precision and recall [Croft et al., 2010]. Its main advantage is summarizing effectiveness in a single number [Croft et al., 2010].

$$F\ Measure = 2 \cdot \frac{(precision \cdot recall)}{(precision + recall)} \tag{19}$$

During the last years, some measures have been widely used, especially mean average precision (MAP) [Manning et al., 2008]. This measure is the most standard among the TREC community, providing a "single-figure measure of quality across recall levels" [Manning et al., 2008]. For the set of top k documents retrieved for each query $q_j \in Q$, MAP is defined according to Equation 20.

$$MAP(Q) = \frac{1}{|Q|} \cdot \sum_{j=1}^{|Q|} \frac{1}{|R_j|} \sum_{k=1}^{|R_j|} Precision(R_{jk}) \tag{20}$$

Where R_j is the set of relevant documents retrieved for query q_j, $|R_j|$ is the number of elements of R_j and R_{jk} is the set of ranked results from the top k results of R_j [Manning et al., 2008]. When no relevant document is retrieved, the precision value is equal to 0; for a single query, the average precision "approximates the area under the uninterpolated precision–recall curve"; and for a set of queries, the MAP would be an approximation of the average area under the precision–recall curves of these queries [Manning et al., 2008].

5 Results and Evaluation

The best results obtained through cross validation for each studied method are listed on tables 4.1 and 4.2. Table 4.1 shows cross validation results for the case of not removing stop words and Table 4.2 presents the results obtained with stop words removal.

No Stop Words Removal				
	MAP	Precision	Recall	*F Measure*
BM25	0.6827	0.7774	0.7307	0.6930
TF-IDF	0.6837	0.7405	0.7456	0.6889
LM	0.6086	0.7923	0.6483	0.6543

Table 4.1: Results for the experiments performed without stop words removal for Okapi BM25 (BM25), TF-IDF and Language Model (LM) approaches.

Stop Words Removal				
	MAP	Precision	Recall	*F Measure*
BM25	0.7136	0.7563	0.7715	0.7107
TF-IDF	0.7259	0.7388	0.7943	0.7251
LM	0.6064	0.8219	0.6365	0.6620

Table 4.2: Results for the experiments performed with stop words removal for Okapi BM25 (BM25), TF-IDF and Language Model (LM) approaches.

As shown in Table 4.1, the best result for the experiments made without stop words removal was achieved for scenario where we use BM25 approach (*F Measure = 0.6930*). For the experiments in which we performed stop words removal, the best result was obtained with TF-IDF approach (*F Measure = 0.7251*).

While comparing experiments made with and without stop words removal, we could observe that the removal of stop words improves the performance of all approaches, especially TF-IDF. In this case the improvement was superior than 5%, if compared to the result of TF-IDF without removing stop words. This could be explained due to the fact that stop words have little value and could also introduce bias since they are high frequency terms. Thus, removing them allow us to focus on the important and meaningful terms.

Our experiments, however, could not surpass the baseline, reported by Adeel Nawab et al. [2012]. The authors achieved *F Measure = 0.882* by comparing n-grams in each document using deletions, WordNet and paraphrases through Language Model approach. Other approaches evaluated in [Adeel Nawab et al., 2012] also "demonstrate that the various types of modified n-grams all contribute to identifying when text is being reused since they capture different types of rewrite operations". Besides that, the authors also show an improvement in the performance when n-grams and Language Model are combined.

6 Conclusion

The goal of this work was evaluate how the classical Information Retrieval techniques perform text reuse task. We made different experiments on METER corpus, studying three classical approaches (Okapi BM25, TF-IDF and Language Model) and compared them to the baseline.

The comparison shows that our results are not good as the baseline and the results obtained by Adeel Nawab et al. [2012]. This demonstrates the relevance of using n-grams techniques in text reuse problems. We could also conclude that it is important to observe text operations like deletions and paraphrases. Both of them are operations generally applied to modified texts.

As future work, we suggest the use of Okapi BM25 and TF-IDF approaches combined with n-gram based techniques. We also recommend the study of other types of text modification and how they could influence the detection of text reuse.

Acknowledgments

We would like to acknowledge the support from GDRI-Web Science and CNRS.

References

Rao Muhammad Adeel Nawab, Mark Stevenson, and Paul Clough. Detecting text reuse with modified and weighted n-grams. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 54–58. Association for Computational Linguistics, 2012.

Ricardo Baeza-Yates and William Bruce Frakes. *Information retrieval: data structures & algorithms*. Prentice Hall, 1992.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

Alberto Barrón-Cedeño. On the mono-and cross-language detection of text reuse and plagiarism. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 914–914. ACM, 2010.

Michael Bendersky and W Bruce Croft. Finding text reuse on the web. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 262–271. ACM, 2009.

Olinda Nogueira Paes Cardoso. Recuperação de informação. *Lavras,[sd]*, 2000.

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical report tr-10-98, Harvard University, 1998.

Paul Clough. Measuring text reuse in the news industry. In Lionel Bently, Jennifer Davis, and Jane C Ginsburg, editors, *Copyright and Piracy: An Interdisciplinary Critique*. Cambridge University Press, 2010.

Paul Clough, Robert Gaizauskas, Scott SL Piao, and Yorick Wilks. Meter: Measuring text reuse. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 152–159. Association for Computational Linguistics, 2002a.

Paul Clough, Robert J Gaizauskas, and Scott Songlin Piao. Building and annotating a corpus for the study of journalistic text reuse. In *LREC 2002*, pages 1678–1685. European Language Resources Association, 2002b.

Paul Clough et al. Old and new challenges in automatic plagiarism detection. In *National Plagiarism Advisory Service, 2003; http://ir. shef. ac. uk/cloughie/index. html*. Citeseer, 2003.

Paul D Clough. Measuring text reuse and document derivation. *Postgraduate transfer report, Department of Computer Science, University of Sheffield, UK*, 2001.

W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*. Addison-Wesley Reading, 2010.

Edberto Ferneda. *Recuperação de informação: análise sobre a contribuição da ciência de computação para a ciência da informação*. PhD thesis, 2003.

Robert Gaizauskas, Jonathan Foster, Yorick Wilks, John Arundel, Paul Clough, and Scott Piao. The meter corpus: a corpus for analysing journalistic text reuse. In *Proceedings of the Corpus Linguistics 2001 Conference*, pages 214–223. Citeseer, 2001.

Nevin Heintze et al. Scalable document fingerprinting. In *1996 USENIX workshop on electronic commerce*, volume 3, 1996.

Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

David M. Levy. Document reuse and document systems. *Electronic Publishing*, 6(4):339–348, 1993.

Vijay Kumar Mago. *Cross-Disciplinary Applications of Artificial Intelligence and Pattern Recognition: Advancing Technologies: Advancing Technologies*. IGI Global, 2011.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

Brian Martin. Plagiarism: a misplaced emphasis. *Journal of Information Ethics*, 3(2):36–47, 1994.

James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 2000.

Hermann A Maurer, Frank Kappe, and Bilal Zaka. Plagiarism-a survey. *J. UCS*, 12(8):1050–1084, 2006.

Guilherme NERY, Ana Paula BRAGAGLIA, Flávia CLEMENTE, and Suzana BARBOSA. Nem tudo que parece é plágio: cartilha sobre plágio acadêmico. *Instituto de Arte e Comunicação Social da Universidade Federal Fluminense–UFF, Rio de Janeiro/RJ*, 2010.

Joaquín Pérez-Iglesias, José R Pérez-Agüera, Víctor Fresno, and Yuval Z Feinstein. Integrating the probabilistic models bm25/bm25f into lucene. *arXiv preprint arXiv:0911.5046*, 2009.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

Stephen Robertson, Hugo Zaragoza, and Michael Taylor. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 42–49. ACM, 2004.

Stephen E Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 232–241. Springer-Verlag New York, Inc., 1994.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. Okapi at trec-3. *NIST SPECIAL PUBLICATION SP*, pages 109–109, 1995.

Catarina Silva and Bemardete Ribeiro. The importance of stop word removal on recall values in text categorization. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 3, pages 1661–1666. IEEE, 2003.

Yorick Wilks. On the ownership of text. *Computers and the Humanities*, 38(2): 115–127, 2004.

ChengXiang Zhai. Statistical language models for information retrieval. *Synthesis Lectures on Human Language Technologies*, 1(1):1–141, 2008.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.